FORMAL SPECIFICATION OF INDUSTRY FOUNDATION CLASS CONCEPTS USING ENGINEERING ONTOLOGIES

A Thesis
Presented to
The Academic Faculty

by

Manu Venugopal

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Civil and Environmental Engineering

Georgia Institute of Technology
December 2011

FORMAL SPECIFICATION OF INDUSTRY FOUNDATION CLASS CONCEPTS USING ENGINEERING ONTOLOGIES

Approved by:

Dr. Jochen Teizer,
Advisor, Committee Chair
School of Civil and Environmental Engineering
Georgia Institute of Technology

Professor Charles M. Eastman, Advisor
Colleges of Computing and Architecture
Georgia Institute of Technology

Dr. Ioannis Brilakis
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Lawrence F. Kahn
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Daniel Castro-Lacouture
School of Building Construction
Georgia Institute of Technology

Dr. Rafael Sacks
Faculty of Civil and Environmental Engineering
Technion, Israel Institute of Technology

Date Approved: September 15, 2011

To my parents, Parvathi and Venugopal.

ACKNOWLEDGEMENTS

Building Information Modeling (BIM) is one of the best things to have happened to the Civil Engineering community in the past decade. BIM is still in its nascent stage and this dissertation which focuses on the topic of interoperability of BIM systems would not have been possible without the help of many people.

First and foremost, I would like to thank my advisors, Professors Jochen Teizer and Charles (Chuck) Eastman. Jochen gave me the opportunity to come to Georgia tech for pursuing my doctoral studies and introduced me to the fascinating world of research involving emerging technologies for construction engineering such as laser scans, sensors and virtual construction at his Rapids Lab. With his attention to detail and quality of work, I learned the basics of conducting serious research from Jochen. I am also greatly indebted to Chuck, the guru of BIM, for taking me as his protégé and giving directions, which only he could have done with his vast knowledge in the area. This research would not have been possible without the untiring and selfless dedication of Chuck towards an open BIM world.

Another person who has contributed a lot through his ideas and detailed feedback is Professor Rafael Sacks at Technion. With his experience in the fields of information technology, virtual design and construction; the feedback and critique given by Rafael has helped to improve this thesis considerably. He also graciously agreed to be on my guidance committee. I would like to thank the other members of my doctoral committee, Professors Lawrence F. Kahn, Ioannis Brilakis, and Daniel Castro for the time and effort they took in guiding me. I was also involved in the development of a National BIM Standard for the Precast Concrete Industry. The work I did as part of this project helped me in formulating the research questions. I am grateful to the NBIMS team whose weekly conference calls including Chuck, Rafael, Ivan Panushev and Shiva Aram, acted as the sounding board for many different ideas and intellectual discussions about interoperability, BIM, IFC, and the latest topics in the computing area. Email and web correspondence with Richard See, Thomas Liebich and

Stefan Richter, who are all experts in the area of interoperability, helped in clarifying some of the concepts and doubts that arose at various stages of my research. The technical discussions and brainstorming sessions I had with Chuck, Jin-kook Lee and Donghoon Yang, while writing proposals for National Science Foundation were of tremendous value. These discussions helped in expanding my horizon of knowledge in the field. I thoroughly enjoyed the time spent working together with construction engineering students at Rapids Lab, especially Tao Cheng, Jeff Bohn, Ben Allread, Uday Mitra, and Sijie Zhang among others. Some of the field trips and team outings we undertook brought us together as a team. The week we spent together collecting data at a capital-intensive power plant construction site in the east coast for a Construction Industry Institute (CII) project immediately comes to mind.

I would like to thank all the faculty and staff at the Department of Civil Engineering, College of Computing and College of Architecture at Georgia Tech. The high standards, demanding course work and excellent lab facilities and interaction with great minds provided the necessary fuel for pursuing a doctoral degree. It would be a great injustice if I do not mention the help and preparation that faculty and staff at National Institute of Technology (NITK), where I did my undergraduate education, provided in creating the foundation for what I am today in terms of an engineer, researcher, and human being. I would like to express my gratitude to my advisors and mentors at NITK, especially Professors Vargheese George, Jayalekshmi, and Narasimhan who have played a major role in inspiring a young engineering student and taught him that knowledge is wealth.

Funding is an integral part of performing any research and I would like to extend my gratitude to National Institute of Standards and Technology (NIST), Charles Pankow Foundation, Precast/Prestressed Concrete Institute (PCI) and Georgia Tech who provided me with the resources for pursuing research. The folks at Tekla also helped by providing support in the form of training sessions and educational licenses to their software.

Studying in a top institution like Georgia Tech is very demanding and one cannot survive without having friends. I consider myself extremely lucky to have known a group of people whom I can call friends. Spending time with Karthik, Shiva, Konduri, Maha, Preethi,

Deepa, Andy and Sasi helped in relaxing and unwinding after a tough workweek. We were more like a family spending so much time talking, playing games and also studying. I could not have asked for better roommates than Kambly and Satheesh whom I knew for a long time. We learnt together, the basics of cooking and always had a good time together. The coffee breaks I used to have with Sandhya always helped as a refresher while working in the lab. The help Sandhya and Maha provided by proof reading different versions of this thesis is immeasurable. Samarth, Lakshmi, Rama, Bhargav, Ranjeeth and Aswhini were instrumental in me feeling at home during my initial days at Georgia Tech. The cricket team we formed at Georgia Tech helped in following one of my passions along with making many great friends. We all had a great time participating occasionally in cricket tournaments in Atlanta. This journey well and truly started from my NITK days where I learned so much outside the classrooms in company of Bijith, Renjith, Ajay, Sam, Guru, Madhavan and Sidharth,

I come from a family where education was held in high esteem. My Achema was always a source of inspiration and constant motivation with her work ethic and unwavering spirit. My parents, Venu and Parvathi, my brother Yadu, and my better half, Shweta supported me in all my endeavors. Pursuing my doctoral study and writing this thesis would not have been possible without the sacrifices and efforts they and many others made for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS OR ABBREVIATIONS

AASHTO	American Association of State Highway and Transportation Officials.
ACI	American Concrete Institute.
AEC	Architecture, Engineering, and Construction.
AEC-FM	Architecture, Engineering, Construction, and Facility Management.
AIA	American Institute of Architects.
API	Application Programming Interface.
BIM	Building Information Modeling.
BPMN	Busines Process Modeling Notation.
B-REP	Boundary Representation.
CAD	Computer Aided Design.
CAE	Computer Aided Engineering.
CAM	Computer Aided Manufacturing.
CICS	Construction Information Classification Systems.
CIS2	CIM Steel Integration Standards Release 2.
CSG	Constructive Solid Geometry.
CSI	Construction Specification Institute.
DWG	Drawing File Format.
DXF	Drawing Exchange Format.
EM	Exchange Model.
ER	Exchange Requirement.
FM	Facility Management.
GSA	The United States General Services Administration.
GUID	Globally Unique Identifier.
HTML	Hyper Text Markup Language.
IAI	International Alliance for Interoperability.
IDDS	Integrated Design and Delivery Solutions.

IDM	Information Delivery Manual.
IFC	The Industry Foundation Classes.
IFD	International Framework for Dictionaries.
IGES	Initital Graphics Exchange Specification.
ISO	International Organization for Standardization.
ISO/PAS	International Organization for Standardization - Publicly Available Specification.
IT	Information Technology.
MVD	Model View Definition.
NBIMS	National Building Information Modeling Standard.
NIBS	National Institute of Building Sciences.
NIST	National Institute of Standards and Technology.
NURBS	Non Uniform Rational Basis Spline.
OMG	The Object Management Group.
OWL	Web Ontology Language .
PCI	Precast/Prestressed Concrete Institute.
RDF	Resource Description Framework.
RDFS	Resource Description Framework Schema .
SDAI	Standard Data Access Interface.
SEM	Semantic Exchange Module.
STEP	Standards for the Exchange of Product Model Data.
SQL	Structured Query Language.
UML	Unified Modeling Language.
VDC	Virtual Design and Construction.
VRML	Virtual Reality Modeling Language.
W3C	World Wide Web Consortium.
XML	Extensible Markup Language.

SUMMARY

Architecture, Engineering, Construction (AEC) and Facilities Management (FM) involve domains that require a very diverse set of information and model exchanges to fully realize the potential of Building Information Modeling (BIM). Industry Foundation Classes (IFC) provides a neutral and open schema for interoperability. Model View Definitions (MVD) provide a common subset for specifying the exchanges using IFC, but are expensive to build, test and maintain. A semantic analysis of IFC data schema illustrates the complexities of embedding semantics in model views. A software engineering methodology based on formal specification of shared resources, reusable components and standards that are applicable to the AEC-FM industry for development of a Semantic Exchange Module (SEM) structure for IFC schema is adopted for this research. This SEM structure is based on engineering ontologies that are capable of developing more consistent MVDs. In this regard, Ontology is considered as a machine-readable set of definitions that create a taxonomy of classes and subclasses, and relationships between them. Typically, the ontology contains the hierarchical description of important entities that are used in IFC, along with their properties and business rules. This model of an ontological framework, similar to that of Semantic Web, makes the IFC more formal and consistent as it is capable of providing precise definition of terms and vocabulary. The outcome of this research, a formal classification structure for IFC implementations for the domain of Precast/ Prestressed Concrete Industry, when implemented by software developers, provides the mechanism for applications such as modular MVDs, smart and complex querying of product models, and transaction based services, based on the idea of testable and reusable SEMs. It can be extended and also helps in consistent implementation of rule languages across different domains within AEC-FM, making data sharing across applications simpler with limited rework. This research is expected to impact the overall interoperability of applications in the BIM realm.

CHAPTER I

INTRODUCTION

This chapter introduces the area of Building Information Modeling to the readers. The motivation behind this PhD research is explained, followed by a brief definition of the problem. Then the research scope and contributions are stated. A brief outline of the thesis is also provided to help the readers understand the flow of the thesis.

1.1 What is Building Information Modeling?

BIM: The term Building Information Modeling (BIM) has come to be associated with the digital representation of the physical and functional characteristics of a facility and also the process of creating, using, and maintaining such a shared knowledge resource as a tool for decision making throughout the lifecycle of a facility [92, 35].

In this research, BIM is considered as the integration of these two ideas of a *process* and also the *digital representation* of a *shared knowledge resource*.

1.2 Motivation and Problem Definition

The motivation behind this research is to improve the interoperability of the BIM applications in the AEC industry.

Interoperability: Interoperability is defined as the ability of diverse systems and organizations to work together or interoperate.

The problem of interoperability in BIM landscape is well documented [35] and is estimated to be costing the industry 15.8 billion dollars every year [43]. Eight in ten users of BIM software tools in the United Stated consider interoperability or lack of it between software

applications to be the limiting factor in achieving the full potential of BIM [124].

The Industry Foundation Class (IFC) schema is accepted as the industry standard for interoperability [67, 69] and is currently in the process of becoming an official International Standard (ISO/IS 16739) [22]. However, model exchanges based on IFC are still error prone and incomplete [75]. The errors are a result of software translators, even those which are IFC compliant, exporting specific data needed for the targeted exchange in a manner semantically not understood by the importing application. Hence, object schemas such as IFCs are a necessary but not a sufficient condition for achieving robust data exchanges. Based on varying exchange requirements, different research and development groups propose model views definitions (MVD), as a solution for specifying exchange requirements. However, the current model view development methodologies, which are based on *usecases* leaves scope for different interpretations based on end-user requirements and lacks a formal framework. Moreover, the granularity and atomicity with which such model views are defined is not consistent across the industry [117]. This adds to the overhead for software developers and hinders IFC based implementations [33]. Hence, there needs to be a way to consistently specify IFC implementations based on exchange requirements. In order for that to happen, additional levels of specificity are required to define model exchange requirements and model views in a formal, consistent, modular and reusable manner.

1.3 Research Questions

The central theme of this thesis is:

How can an improved model view development methodology be supported that reduces the inconsistency and redundancy of model views developed by parallel development teams, thereby improving the overall interoperability of BIM tools?

Three major research questions raised in this research and investigated are as follows:

1. **How can we develop model views consistently across research teams and domains?**

   In order to support IFC implementations, the consistency of model views designed is

an important criteria. Lack of which causes an overhead to software developers and inhibits new IFC implementations.

2. **What should be the building blocks of model views for semantic information exchanges?**

   The current approaches to model view development creates redundant information that is spread across several domains due to lack of reusability. Defining the building blocks of model views and packaging them in an object-oriented, modular and reusable manner is needed. This leads us to the third question.

3. **What are the semantics of model views for information exchanges using the IFC schema?**

   There is a need to analyze the complexities of embedding semantic meaning in model exchanges using the IFC schema. Such an analysis can provide insights into the structuring of information items for model view development work.

1.4 Contributions

This thesis aims to improve the interoperability of BIM tools in the AEC-FM domain in two ways:

1. By providing a formal definition of IFC entities, relations, attributes, and constructs for information exchange, using engineering ontologies and packaging them into modular units.

2. By improving the model view development methodology into an object-oriented, consistent and modular process.

Semantic Exchange Module (SEM): An SEM is a structured, modular subset of the objects and relationships required in each of multiple BIM exchange model definitions.

The contributions are explained in the following paragraphs.

- Semantic Exchange Modules (SEM) are introduced as the building blocks of future model views. Its raison d' être is to enable BIM software companies to code, import

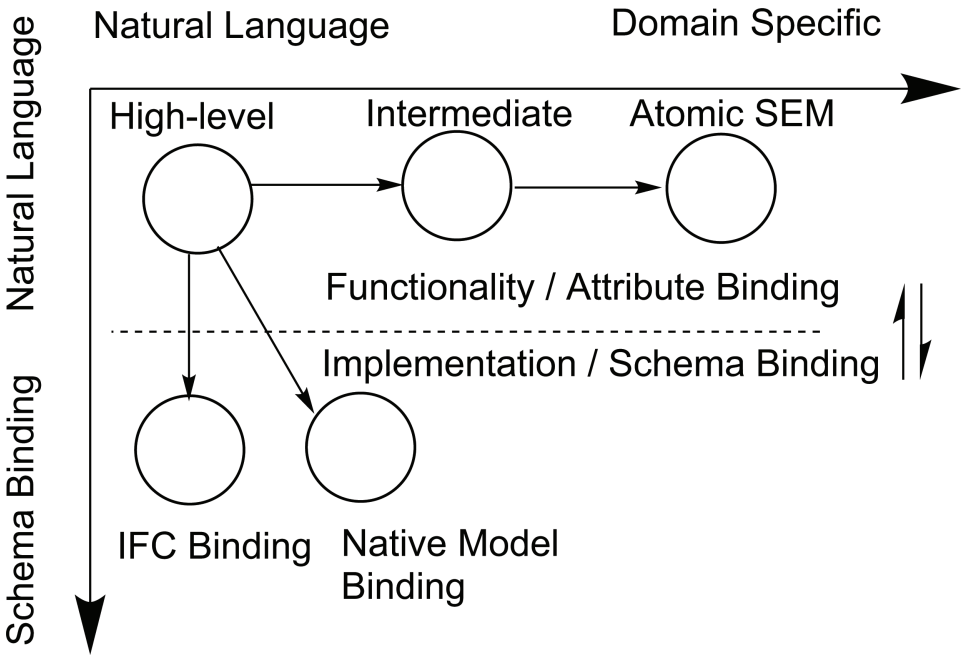

Figure 1: Two dimensional structure and mapping of Semantic Exchange Modules (SEM).

and export functions in modular fashion, such that a function written to export or import model objects according to any given SEM can be tested and certified once, and then re-used to fulfill multiple exchange model exports/imports without modification. A SEM graph, usually has two dimensions as shown in Figure 1. The first dimension is the classification hierarchy of different entities involved and their relationships. The second dimension involves the implementation of each of these nodes in the graph by mapping it to a schema (IFC and native). The branches of this dimension represent the data access paths. Therefore, a SEM has:

1. a definite mapping to a schema,
2. mappings to a native model (when fully defined),
3. methods to map between the two bindings (IFC and native),
4. data access paths and
5. belongs to a specific classification hierarchy.

- A semantic analysis of IFC schema provides insights into the complexities of embedding information in model exchanges. Some of the issues highlighted are type-instancing, classification schemes, geometry, relationships and rules. A set of guidelines are provided to improve the consistency of model views and SEMs. A minor objective of this analysis is to provide recommendations to adapt the IFC structure to ISO standards by elaborating on the ambiguities and inconsistencies that became visible as a result of this research.

- Engineering ontologies are used to define the model views in a formal and reusable structure. An ontology is a formal representation of an abstract, simplified view of a domain that describes the objects, concepts and relationships between them that holds in that domain [50]. The scope of this work is to identify the semantics that are implicit in the IFC schema, keeping release 2x3 as a reference. It is a very large task to define model views that are required for the whole domain of AEC. To date, the precast concrete industry is one of the most advanced domains, as no other set of exchange models have been fully specified following the NBIMS procedures. Therefore, precast concrete has been selected as the scope and the research is restricted to those entities, relations and attributes that are necessary to define a precast concrete exchange standard.

- A new approach of defining model views based on SEMs is introduced. Each SEM is defined as a modular unit, which is unit tested for completeness. Defining a model view is reduced to plug-and-play of SEMs from a predefined library. This eases the load on testing and validation as the model views are built from already tested SEMs. It is envisioned that by following this methodology, the time and effort required for a new IFC implementation can be reduced.

1.5 Organization of the Thesis

This thesis describes an investigation into the use of ontologies for formal specification of Semantic Exchange Modules and their suitability for defining model views. The following is an outline of how this thesis is set-up.

Chapter 1 introduces the field of Building Information Modeling as a process rather than just a software or 3D model. The motivation for this research is to improve the interoperability of application in the BIM arena. Industry is facing the issue of model exchanges being incomplete/error-prone due to the lack of semantic definitions within IFC. Hence, the objective of this research is to provide a formal definition of IFC entities, attributes, etc. The scope of research is the model exchanges for precast concrete industry.

Chapter 2 gives a brief account about BIM in the AEC-FM industry and the evolution of the IFC as an industry standard. The efforts of the National BIM Standard in describing the processes in this aspect are reviewed before explaining the focus area of this research. The gaps in current interoperability research are summarized at the end of this chapter.

Chapter 3 introduces the topic of knowledge representation systems. A literature review about how ontologies can be used to describe knowledge in a formal way is provided. Some parallels between the research objectives of other knowledge sharing paradigms and this research are investigated. Research in the AEC-FM industry focusing on this area is discussed in the summary of this chapter.

Chapter 4 presents the hypothesis and explains the research methodology of developing an ontological foundation for IFC and packaging the knowledge into a graphical structure for SEMs, which can be tested once and reused.

Chapter 5 explains the results of a rigorous analysis of IFC schema and its suitability for embedding semantic meaning in model exchanges. Special focus is given on issues such as type-instance hierarchies, classification schemes, geometry, relations, rules, etc. The results are grouped in the form of recommendations to improve the IFC ambiguities, current model view approach and also for SEMs.

Chapter 6 introduces the assumptions made for the ontology definitions and explains the process of developing an application ontology for precast piece.

Chapter 7 introduces the requirements for the SEMs based on the semantic analysis results and explains the definitions of the SEM structure based on an application ontology. The result of this chapter provides example SEMs for use in defining model views using the IFC schema.

Chapter 8 provides a proof of concept in the form of validation and verification of the ontology-based concept structures developed and Semantic Exchange Modules. Test cases for verifying different aspects of the research are discussed with results. A Model View Developer plugin for BIM tools is written in C# that makes use of the SEM libraries. This allows the users to compile a new model view in a modular and reusable structure. A test bed model is developed with standardized precast pieces and profiles in order to validate the exchanges in general and the fabrication model exchanges in particular such that the procurement stage of a precast project can be automated.

Chapter 9 discusses the impact of this research in developing robust and consistent model views and also in easing the testing and validation requirements for IFC implementations. Also discussed is the IFC target of becoming a full International Standard and how this research identifies some potential ambiguities. A future extension of this research in the form of an automated model exchange protocol is also discussed. Chapter 10 discusses the conclusions of this thesis and also summarize the findings.

Appendix A introduces the Precast National BIM Standard project. The Information Delivery Manual (IDM) developed as part of the precast concrete National BIM Standard is presented. This is taken as the scope for this research.

Appendix B contains the OWL specifications for the Precast System ontology and Appendix C contains the Model View Definition specified in EXPRESS format for the Precast System. Appendix D contains the test bed model with precast pieces for validating exchanges.

CHAPTER II

BACKGROUND

The past decade has seen the use of BIM as a central information management paradigm in AEC projects gain popularity. The major factor influencing the value proposition of BIM in industry is the efficiency of interoperability solutions that provide flawless streamlined information flow between different disciplines in a project. The Industry Foundation Classes (IFCs) provide the means for interoperability. However, IFC implementations in software need clear guidance for specific purposes and projects. This chapter deals with interoperability and the evolution of IFC. It also explores the National BIM Standard initiative. A summary of the current approaches in the industry and its drawbacks are also addressed.

2.1 Overview of BIM in AEC-FM industry

BIM tools serving the AEC-FM industry cover various domains and have different internal data model representation to suit each domain. There is no one single application that can provide the entire set of functionalities required for the AEC-FM industry [35]. Yet building design and delivery calls for building information modeling data to be exchanged between various actors in order to integrate the various types of expertise needed to realize the overall project. The AEC industry requires rich and varied tools to satisfy requirements in architectural and structural design, structural analysis, space planning and energy analysis, 4D simulations, work task allocations, etc. Table 1 lists some of the BIM tools currently in use for different application areas and also the different data formats in use. Hence, this raises the issue of compiling a coherent information model of the building.

Table 1: Example of the multitude of BIM tools and diverse data formats in use.

Feature List	Details
A. BIM Tools	
Massing	Sketch up, bonzai3D, Rhino
Design	ArchiCAD, Revit, Tekla, Bentley Architecture, Vectorworks
Construction Management	Constructor, Vico, Tekla, Navisworks, Synchro
Cost Estimation or Specification Tools	DProfiler, US Costs, Innovaya, Prolog
Structural Analysis	SAP, ETABS, GT STRUDL
Energy Analysis	Energy Plus, DOE-2, Equest
B. Data Formats	
3D Surface and Shape Format	3DS, WRL, STL, IGS, SAT, DXF, DWG, OBJ, DGN, PDF(3D), SGL, DWF, U3D, IPP, PTS
3D Object Format	STP, EXP, CIS/2, IFC
XML Format	IFCXML, AecXML, Obix, AEX, bcXML, AGCxml

2.2 Evolution of Industry Foundation Classes

Software industry economics is based on competition and survival in this highly competitive environment has caused companies base their data formats native to their application and proprietary. Hitherto, lack of interoperability is an issue restricting BIM applications from attaining their full potential. Figure 2 shows the timeline of various activities that relate to CAD and related technology scenario. Data exchange problems have existed since the early stages of CAD [103] and there have been various approaches to overcome these problems. The costly method is required to custom-build translators specific to applications. As an alternative, neutral file formats such as DXF, IGES and SAT were developed to facilitate file exchange between one system and another. However, the information content of these files were limited to geometric entities such as lines, arcs, planar surfaces, curved surfaces, primitive solids, etc. Manufacturing Industry pioneered the development of the product model exchange standard in ISO-STEP (Standard for the Exchange of Product model data) in the late 1980s. EXPRESS language was developed with a set of libraries of re-usable structures and methods, enabling the transition to object-based exchange models.

The need for an open standard specifically for the AEC-FM industry was recognized and Industry Foundation Classes (IFC) were started in the year 1994 as a result of an industry led consortium called Industry Alliance for Interoperability. This was later changed to International Alliance for Interoperability (IAI) in 1997 and now BuildingSMART. IFC represents geometry, relations, processes and material, performance, fabrication and other properties, needed for design and production, using the EXPRESS language [82].

Industry Foundation Classes (IFC): IFC is a neutral, open and object-based standard (ISO 16739) that provides a data model schema for interoperability by supporting comprehensive specification of information throughout the AEC-FM project lifecycle, globally, across disciplines and software applications.

Interoperability enhancement requires common understanding of industry processes as well as the information required for and resulting from executing these processes [122]. IFC exchange implementations enable software developers to access the highly structured and attributed object model data used in Building Information Modeling through an Application Programming Interface (API). Implementations achieve data sharing by using the IFC and Standard Data Access Interface (SDAI) [68]. Figure 3 shows the architecture diagram of IFC. The different domains served by the IFCs are shown and also the different resources providing necessary functionality. Due to the breadth and flexibility offered by its coverage, IFC can be called a rich product-modeling schema. However, IFC does not define the specific information exchange requirements for users. This makes it possible for two applications to export or import different information while describing the same object. Hence, the degree of success in exchanging 3D model-based data using IFC will depend on the performance of the export and import translator functions embedded in the BIM tools to convert data from native format to IFC and reverse, based on a standard protocol. Thus, data exchanges are often unreliable due to inconsistencies in the assumptions that different implementers of exchange functions make about how information should be expressed [106]. There are often unpredictable ways in which export and import functions treat the same data, posing a barrier to the advance of BIM [36, 97]. Moreover, when the internal data

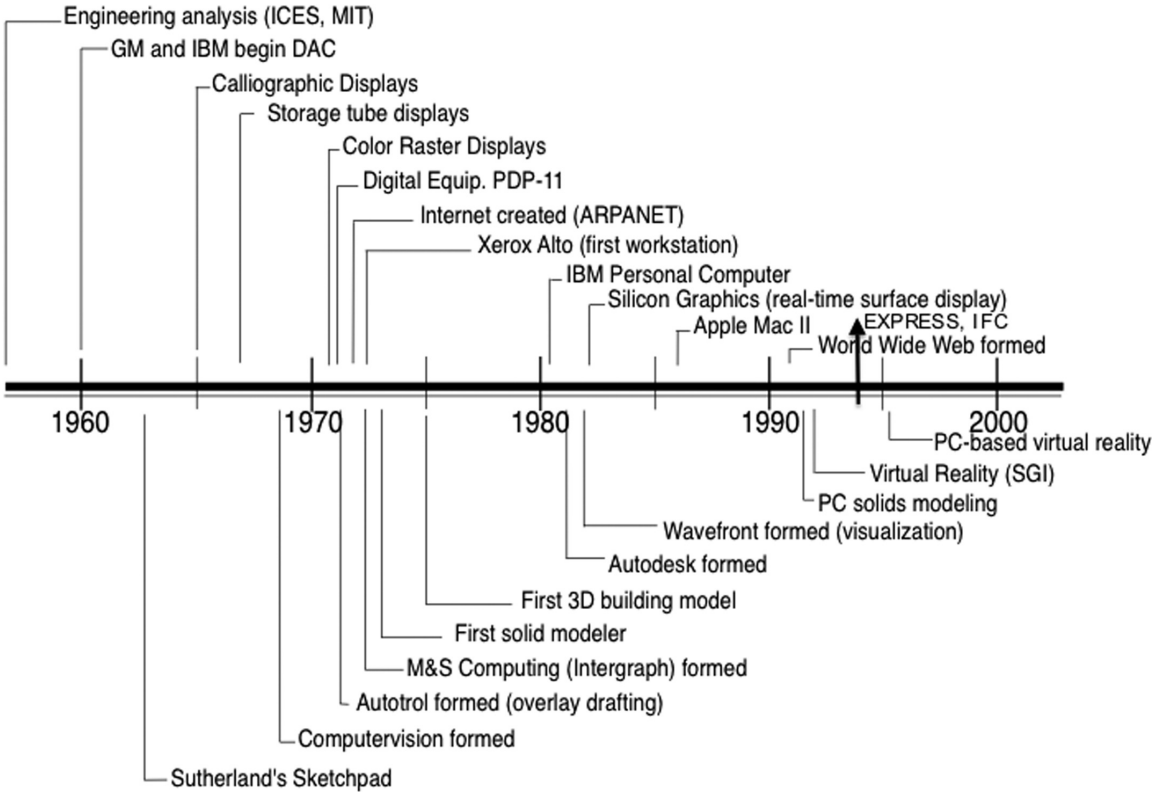

Figure 2: A timeline of CAD and technology initiatives [38].

structure is different and the range of data objects to be communicated is varied, there is ambiguity in how each application defines this. Benchmark tests conducted on exchange of modeling data [70] have shown that since the BIM tools have different semantically defined objects and the users of BIM tools modeled them differently, the IFC file exported from each BIM tool was mapped differently into IFC. These anomalies have led to a conclusion that the domain-specific Model View Definitions (MVD) are needed to define precisely how the building model exchanges should be expressed using the IFC data schema [60].The National BIM Standard is an effort aimed at standardizing the data required for particular workflow exchanges [92].

2.2.1 Test Case: Model Exchange using IFC schema

To illustrate the model exchange scenario between two BIM applications, a real project was considered. The project selected was the G. Wayne Clough Undergraduate Learning Commons building at Georgia Institute of Technology. This was a $93 million building with

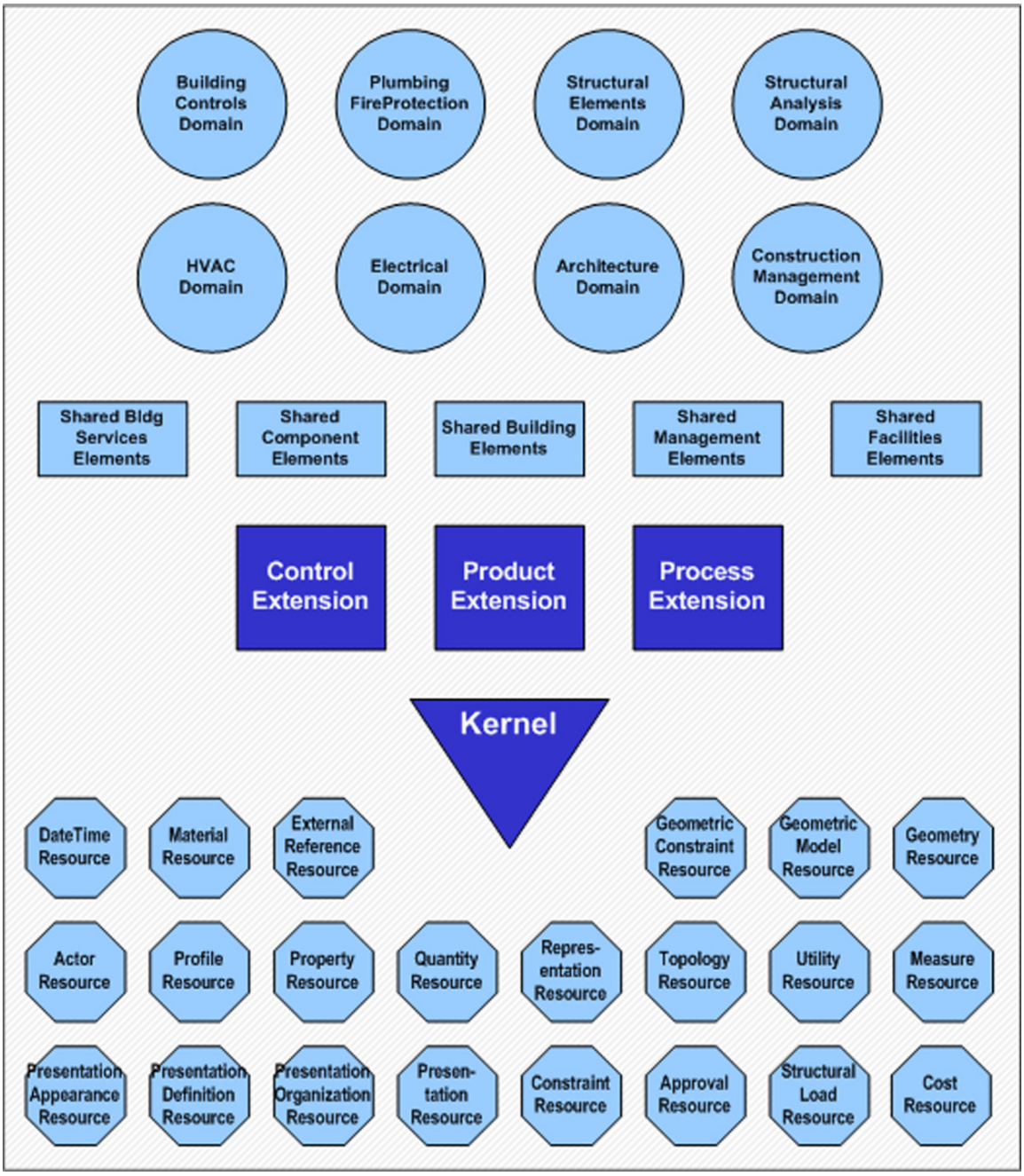

Figure 3: Architecture diagram of IFC schema showing the domain and resource structure [67].

220,000 square feet built-up area. A model was created in REVIT (Figure 4 - a), but the same information had to be recreated for detailing, MEP, etc. by subcontractors. This model was exported into IFC using the existing IFC export function in REVIT. The same was imported into TEKLA but the results showed deficiencies in the export and import functions. Geometry translators are matured enough, but the imported model in TEKLA did not contain the relative placements or the correct object type information. The assembly information was also lost in the exchange process. Figure 4: (a - c) shows the native model in REVIT, the geometry only model in open IFC viewer and the actual building under construction, respectively. To summarize:

1. Beams, columns, walls, footings and stairs were successfully exported.
2. Opening elements, plates and discrete accessories, connections etc. were missing in the export.
3. Spatial hierarchy was not exported.
4. Similarly, assembly information was missing.
5. All geometry was in the form of faces (faceted B-Rep).
6. Placement information of objects were partially exported (Relative placement of some entities were wrongly translated).

2.3 The National BIM Standard

The National BIM Standard^{TM} (NBIMS) initiative [92] proposes facilitating information exchanges through *Model View Definitions* (MVD) [60]. Interoperability enhancement requires the following;

1. common understanding of industry processes and
2. the information required for and resulting from executing these processes.

All the requirements that satisfy a particular model exchange requirement in the industry constitute a subset of the entire schema, that is, a *model view*. This methodology defines the appropriate information entities from a schema for a particular *usecase*.

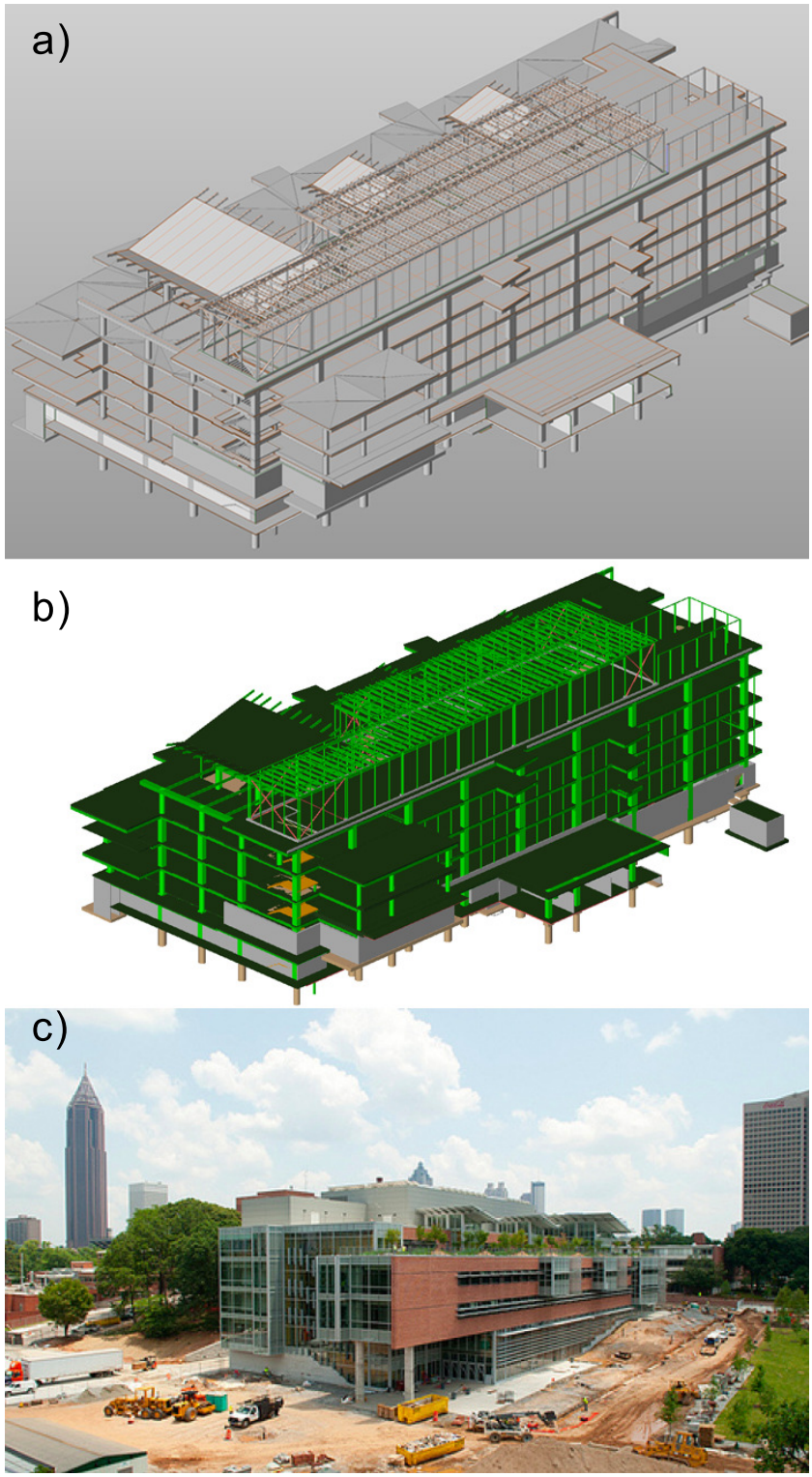

Figure 4: Clough Building Project at Georgia Tech - a) Native model in REVIT, b) Geometry in open IFC viewer, and c) Actual building under construction in Summer of 2011.

2.3.1 Introduction

Building design and delivery calls for building information modeling data to be exchanged between various actors in order to integrate the various types of expertise needed to realize the overall project. There is no one single application that can provide the entire set of functionalities required for the AEC/FM industry [35]. Moreover, product model schemas such as IFC are rich, but highly redundant offering many ways to define objects, relations, and attributes. Hence, effective exchanges require providing a layer of specificity over the top of an IFC (or any other) exchange schema. The purpose of this layer of information is to select and specify the appropriate information entities from a schema, their attributes, and rules governing their possible values for particular uses [92].

Model views are akin to database views that can be defined virtually from specialized and structured subsets of data compiled from populated building models. The selected entities that comprise a model view definition are a subset of all those in the schema. A view can be considered as a query permanently stored within the database and is often implemented in this way. Thus a view is a new model populated with instances derived from the source data model [38]. Views can be classified into two types, first the virtual view, which is stored as a query, and second after derivation is stored as a subschema. We are interested in the second type of definition for model views for IFC. A more generic definition of an MVD is given in [119].

Model View: A model view is a subset of a building product model schema that provides a complete representation of the information concepts needed for a particular information exchange in an AEC workflow.

The notion of a model view based on this definition is shown graphically in Figure 5, where an architectural concept model is passed from an architect to an engineer. The architects or designers will use a BIM authoring application to develop a building model that consists of conceptual layout of building elements into simple assemblies, without surface or structural detailing. Suppose, the purpose of this exchange is to pass this information to an engineer for structural detailing of building elements. The architectural and structural

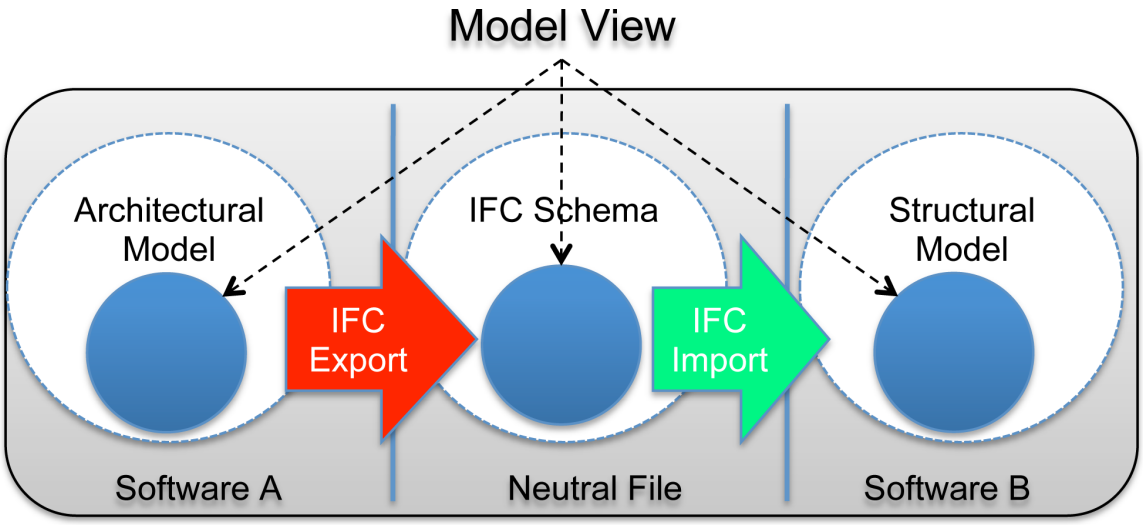

Figure 5: Illustration of Model Views as a subset of the exchange schema.

models will each have details, which are extraneous to this exchange, but there is a subset, which is common to both models, and that constitutes the exchange. Such a subset of the entire IFC model on a project, selected for a particular exchange, is what is defined as a model view in this thesis. Due to the popularity of the IFC, it is assumed as the medium of interoperability for model exchanges. The following sections provide a closer look at the process of defining such views for model exchanges, as described in NBIMS [92].

2.3.2 Overview of the NBIMS Approach

The National BIM Standard Version 1 Part 1 presents a draft of procedural steps to be followed in developing model views. The NBIMS process is shown diagrammatically in figure 6. It outlines thirteen steps in four major phases. A brief overview of this methodology is given in the following paragraphs, but a more detailed review of the process can be found in [36].

Phase 1: Forming a workgroup and identifying the scope and context for one or more use-case exchanges are performed as part of this phase. The workgroup is composed of experts in information technology applications and information exchange for AEC and also experts in the construction domain being considered. The context is defined through compilation of one or more process maps that identify where the exchanges take place in the project lifecycle and who are the actors and applications that are the senders and recipients of the

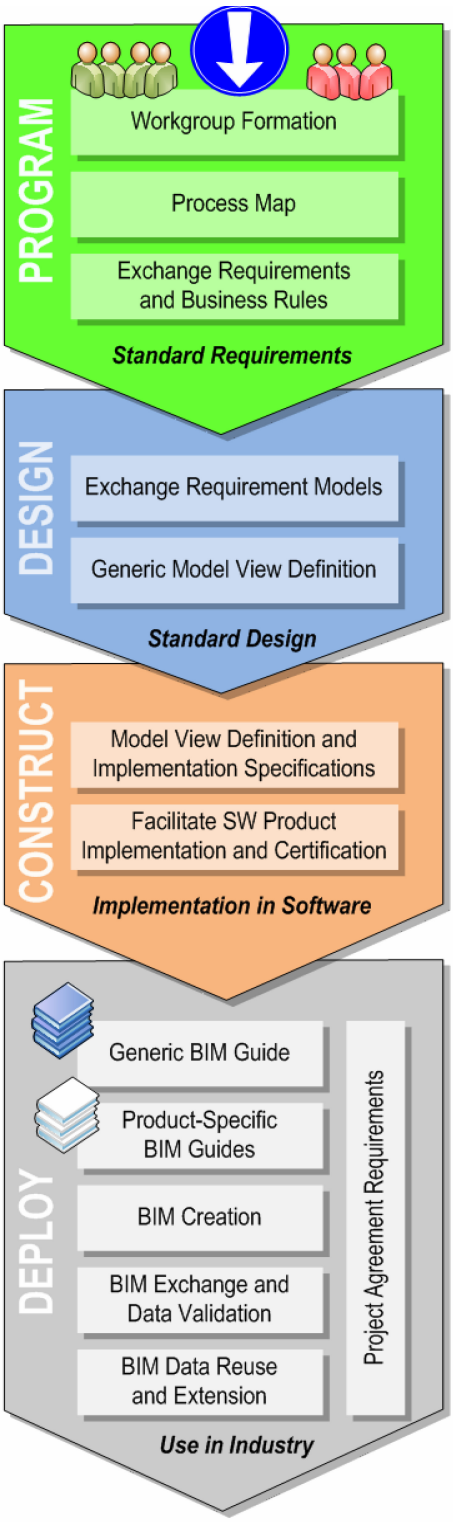

Figure 6: The NBIMS Model view approach [92].

exchange(s). For each exchange, the functional requirements of the information to be exchanged are defined and are called Exchange Requirements. These are structured into what is called an Information Delivery Manual (IDM). The IDM serves as the overall requirement specification for the exchange(s).

Phase 2: The Exchange Requirements identified in the IDM are next structured into a set of information modules. These information units of the exchange are named as MVD concepts (or SEMs, as they are called in this research). A Model View is specified as a collection of such Concepts, which will later be mapped to the implementation schema (IFC most commonly). The input of software developers, representing the major vendors in the domain, is highly desirable in this step. The MVD Concepts are shared through an open website, IFC Solutions Factory [13]. A well-structured set of templates has been developed for documenting Concepts and their aggregation into higher-level Concepts, and then into a Model View for a single exchange. When the templates are filled out, the resulting online documentation serves as the specification for the Model View Definition, which is the second major document in developing a BIM standard. Figures 9 and 10 show a sample MVD and MVD Concept (The SEM defined in this research is a novel idea and is different from the generic notion of concepts as addressed in NBIMS), which are explained later in detail. The validation of MVD can be carried out by its comparison with the IDM.

Phase 3: The third phase addresses the implementation of the Model Views by the domain software companies. Implementation is supported by test cases, for small groups of related MVD Concepts, allowing incremental implementation, testing, and with carefully developed test cases-validation of the data import/export. The Concept level testing can also be called Unit testing. It is an important component of certification.

Phase 4: The last stage involves the development of guidelines for documenting the model views within each of the different supporting applications. This will allow the users of the applications to prepare models suitable for the required exchanges. This phase also includes the development of project models that can be used for real life testing, at the unit and full project level. Test sites that are developed to support certification, are being developed to support interface testing, for both software companies and end users. High confidence in

the ability to exchange design and engineering data reliably is the targeted outcome. The overall process is meant to incorporate best software engineering practices.

Extensive work is done in developing a Precast National BIM Standard [34], by following this methodology and elaborating it in detail and also validating through the lessons learned [33]. Based on this experience, phase two and three of the NBIMS approach are identified, as shown in Figure 6, as the critical aspects in defining a model view. The importance of these two phases is explained in detail in the following section and forms the focus of this thesis.

2.4 Focus of this Research

As mentioned in the previous section, this research focuses on phases 2 and 3 of the NBIMS approach. Concerns about how the exchange requirements are documented and how these are translated into model view definitions arose during the Precast NBIMS project. These concerns and the discussions that followed are the origins of this research. An improved methodology for development of process maps and identifying usecases for model exchange are explained in [3, 36].

Process Maps: Process maps are developed with the goal to analyze the information value chain throughout projects and identify the workflows that are to be supported by BIM tools.

As a part of the precast project, separate process maps were developed based on different project delivery methods. Figure 7 illustrates an example process map and its components [34, 3]. A *Usecase* defines the information exchanges between any two actors in a project aimed at achieving a specific goal, within a specified phase of a project's lifecycle. For example, as shown in figure 5, building information to be exchanged between an architect and an engineer during schematic design is defined as the *usecase*.

Usecase: A usecase defines an exchange scenario between two well-defined roles for a specific purpose, within a specified phase of a building's life cycle.

Model Exchanges should consist of all of the information groups and all of the possible attributes that are needed while making a targeted exchange. The goal of each model exchange

is to support some business activity or domain. To represent the exchange model content, a template was developed.

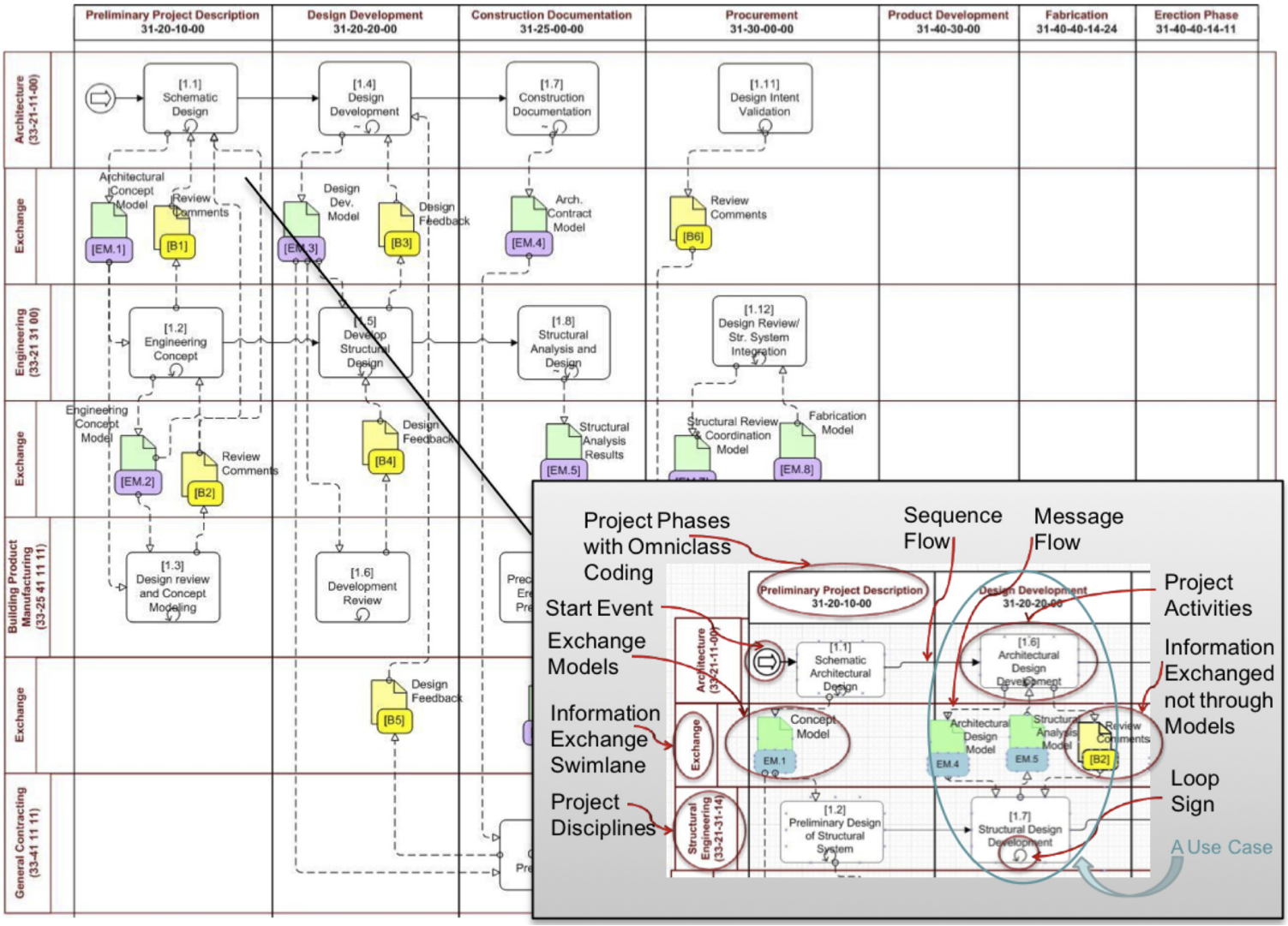

Figure 7: Example process map developed for precast model view definition [34] and illustration of actors, processes, usecase, model exchange, etc [119].

This model exchange template includes:

1. the Omniclass description for project stage and disciplines,
2. a verbal description of the purpose of the exchange, required content, optional content,
3. the example softwares and
4. a listing of other interacting exchanges.

Table 2 shows an example exchange model for a preliminary structural system design. The content of the information exchanges for each usecase is termed *Exchange Requirements* (ER). Figure 8 shows an example. The first column shows the information group that represent the major classes of objects in a building model (e.g. site, buildings, assemblies, foundations, rebar, etc.), followed by a listing of information items for this group, in the second column. The information items provide specific examples of the members of each information groups. These items are described in more detail using a set of attributes. Business rules such as Required (R) or Optional (O), Not needed (blank), etc. are also included, followed by the occurrence of this information group in the various exchange models. Forming a workgroup, developing a process map, defining the set of usecase exchanges being addressed, describing the activities involved and the exchange requirements, together with the exchange priorities are documented in the form of an *Information Delivery Manual* (IDM). The goal of development of an IDM is to capture to the largest extent, the complete requirements of the exchanges. These exchanges are in the form of exchange models that are translated into the appropriate data schema in later implementations of model views. Therefore, IDM can be called as a functional specification for a family of exchanges.

2.4.1 Semantic Exchange Modules (SEM) and Model View Definitions (MVD)

The definition of a model view and the steps leading to the development of such views is explained in detail in the previous sections. The next step in defining a model view is the translation of the exchange requirements (ER) from the textual form so that they can be bound to a particular exchange schema. In this research, this is done by conceptualizing these requirements into reusable modules of information called *Semantic Exchange Modules*

Information Group	Information Items	Attribute Set	Attributes		EM.1	EM.2	EM.3		
Foundations									
	Grade Beam, Pier Cap, Spread Footing, Slab on Grade, Stem Wall, Retaining Wall, Drilled Pier, Cassion, Pile, Pile Cap	Shape	Geometry	Required?	R	R	R		
				Deformations	A	A	A		
				Function?	V	F	E		
				Accuracy?	P	C	C		
			Dimensional Tolerance Info	Required?	R				
		Type	Structural Type (CIP Concrete/Precast)	Required?	R				
		Supplier	GC/Contractor/Fabricator type/name	Required?	R	**Legend**			
		Material	Material type	Required?	R	R - Required, O-Optional			
			Quantity	Required?	R	A - As cast or fabricated, D - Deformed			
		Assembly relations	Part of structural system (slab, floor, façade, frame)	Required?	R	V - Viewable only, F - Reference geometry, E - Editable geometry			
		Nested relations	Contains rebars/tendons…	Required?	R	P - Planar/mesh, C - Curved Surfaces			
			Contains connection hardware….	Required?	R				
		Connection relations	.. to Precast	Required?	R				
			.. to CIP	Required?	R				
			.. to Steel	Required?	R				
		Meta Data	Author, Version, Date	Required?	R				
			Approval Status, Date	Required?	R				

Figure 8: An Illustration of Model Exchange Requirements (ER) collected in a tabular form [119].

Table 2: Exchange model (EM) description for Precast Architectural Concept Model for a preliminary structural system design.

Project Stage	31-20-10-00 Preliminary Project Description
Exchange Disciplines	33-21-00-00 Architecture 33-21-31-14 Engineering 33-25-41-11-11 Building Product Manufacturing
Description	1. Purpose is to define/review basic structural system design, in terms of span directions, vertical transfers,major bracing, foundation pads 2. Architectural concept model involves basic spatial layout of the building including external skin and building shape. It should identify potential locations for vertical load transfers 3. It might involve major architectural finishes, structural system selection, structural grid and site analysis. 4. Revit, Archicad, SDS/2, Tekla, STAAD Pro, RISA, ETABS 5. May be round trip or one-way
Related Exchange Models	A_EM.1, P_EM.1, S_EM.1

(SEM). The idea of creating composable and reusable modules originated from the work done by Professor Charles Eastman, and his PhD students - Manu Venugopal, Jinkook Lee, and Donhoon Yang at the Digital Building Lab @ Georgia Tech towards an *Automated Building Model Exchange Protocol*. This is submitted as a proposal to the National Science Foundation (NSF). The acronym SEM was proposed by Professor Rafael Sacks at Technion University to differentiate it from the different terminologies such as *concept, construct, etc.*, during one of our meetings. A SEM is a structured, modular subset of the objects and relationships required in each one of multiple BIM exchange model definitions. Its raison d' être is to enable BIM software companies to code import and export functions in modular fashion, such that a function written to export or import model objects according to any given SEM can be tested and certified once, and then re-used to fulfill multiple exchange model exports/imports without modification. For example, in the case of precast concrete, the information can be categorized into geometry, material, placement, embeds, connection components, plant and field applied connections, some general attributes, etc. Figure 9

shows a graphical representation of such a model view example. The SEMs are meant to be generic, i.e. they are not specific to any product model schema [60]. Each SEM can be bound to one or more specific schema for implementations. For example, binding to IFC 2x3 entails defining their implementation in terms of IFC entities and relationships. The SEMs and their bindings can be aggregated and published as *Model Views*. This has been done for the precast concrete NBIMS [34]. Software application export and import functions can then be implemented for each exchange by using the SEM bindings as a specification. Figure 10 shows such a binding diagram for Precast Connection Components. The first part shows the details about the SEM, such as title which is a unique identifier, description, IFC release to which the binding conforms, history, references, authors, etc., and usage in view definition. The usage in view definition shows the reuse of this SEM in different places in a model view. The second part shows a sample 3D representation of the same example, followed by the IFC binding relationships. Additional business rules for implementation and an example part-21 file snippet are also provided to help the implementers. The business rules address questions such as, *Are the attributes Required or Optional? Is the attribute referencing a Select or an Enumerated type?* In the latter case, *what are the allowable values? Are there naming conventions to be followed?*, etc. These are called the implementation agreements and the sample Part-21 file is populated with values illustrating these agreements. The current practice of writing translators for exchanges on a case-by-case basis is not efficient and involves much redundancy and overhead. If software companies implement their internal mappings from their own data model to the exchange modeling language organized by SEMs high levels of re-use are possible at the translator writing level. These functions are then tested against the model views, which are themselves supposed to fully represent the semantics of the application requirements captured and specified in the IDM, for validation and certification. The same procedural methodology is followed for all exchanges. For example, we would use the same methodology for a model view for exchange between structural design and structural analysis, or one for structural design to precast detailing, etc.

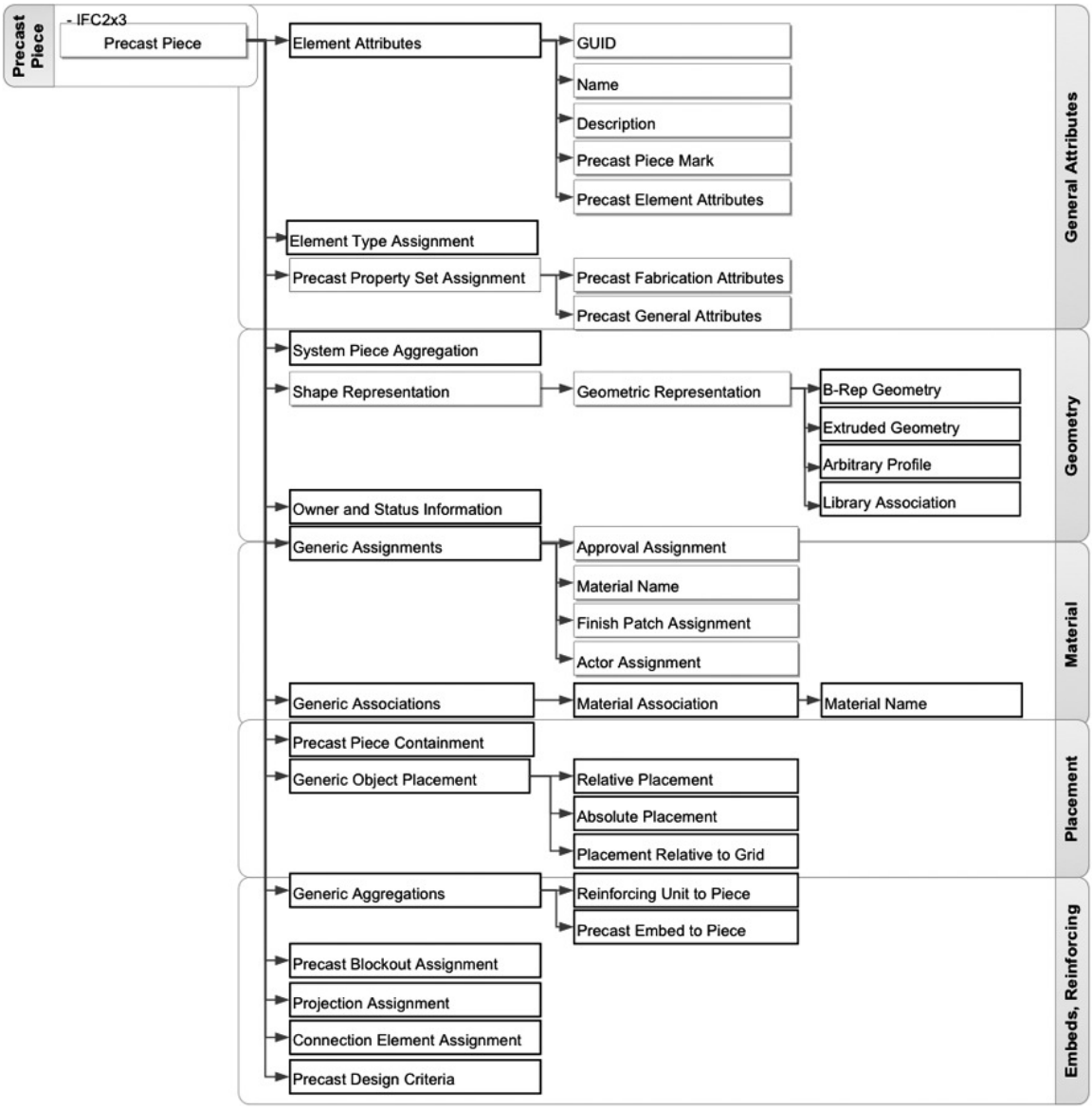

Figure 9: An example for a Model View for precast concrete piece with modules of information or SEMs grouped together [119].

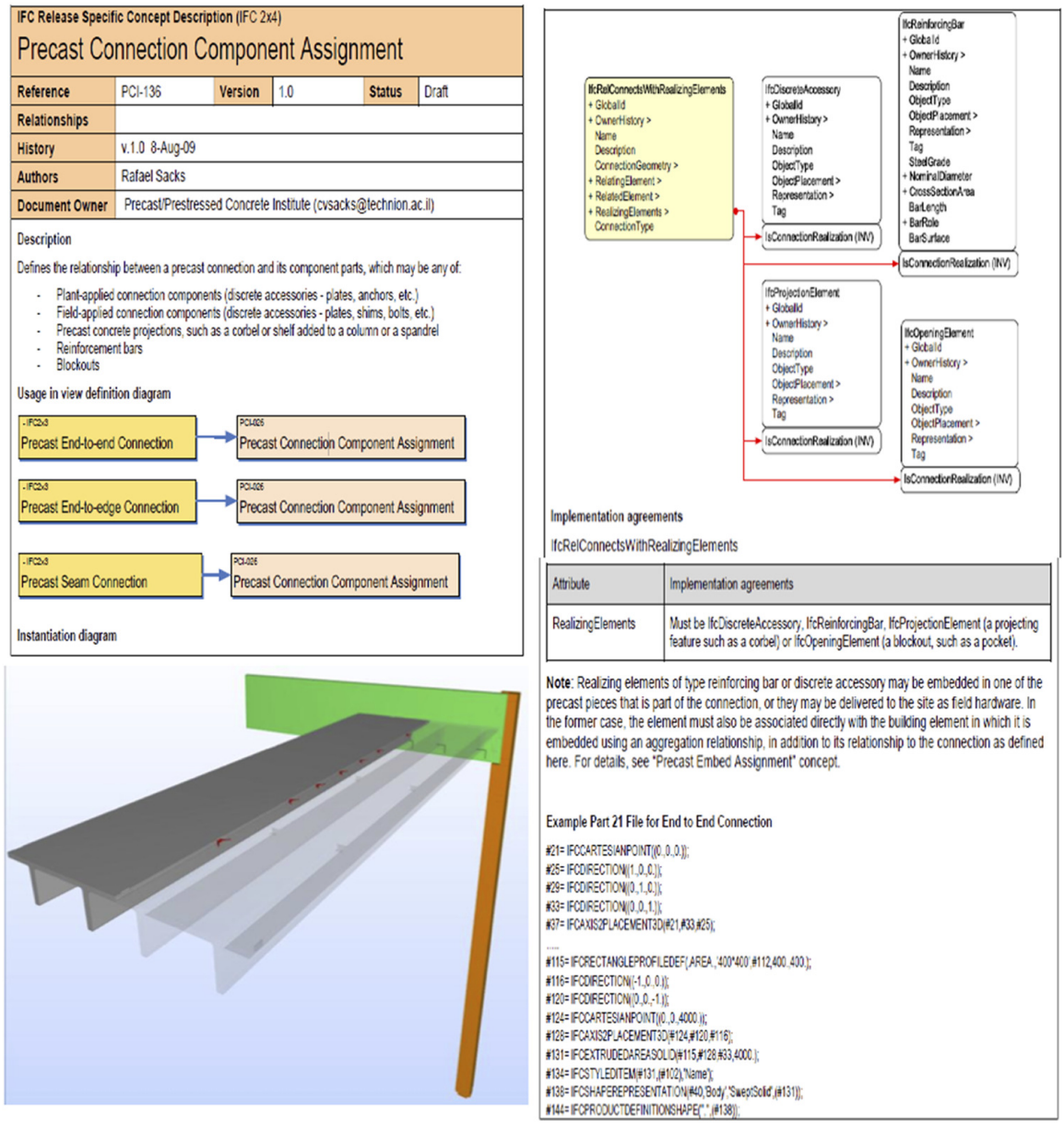

Figure 10: A binding document prepared for precast connection component showing usage info, mapping to IFC, 3D representation, business rules and sample Part-21 file snippet [119].

2.4.2 Summary: Gaps in Interoperability Approaches

IFC is based on EXPRESS, which is a representational language, and STEP definitions. The STEP standard is known to be highly expressive but lacks a formal definition of its concepts [55]. Hence, semantic problems arise when implicit background information about entities is not available. Similar to many data schemas, IFC is highly redundant, offering different ways to define objects, relations and attributes. Thus, data exchanges are not at an acceptable confidence level due to inconsistencies in the assumptions different implementers of exchange functions make about how information should be expressed [106]. There are often unpredictable ways in which export and import functions treat the same data, posing a barrier to the advance of BIM [36, 97].

The National BIM Standard^{TM} initiative (NBIMS) [92] proposes facilitating information exchanges through *Model View Definitions* (MVD) [60]. Interoperability enhancement requires (i) common understanding of industry processes and (ii) the information required for and resulting from executing these processes. The work done on the Precast National BIM standard [34], which is one of the early NBIMS, has given insights into the advantages of the MVD approach. This has enabled identification of areas that require attention, leading to the research presented in this thesis.

Two sets of semantics are at the core of any Model View Specification [118]:

1. the user/application functional semantics defining the information that must be exchanged;

2. the representational semantics available in IFC or other data modeling schema for representing the user intentions.

Any person defining models in IFC (or other schema) asks and resolves the following example types of questions.

How does one represent in IFC:

- type-instance relations

- shape families (may be different than type instance)

- patterns of layout, such as rebar, tiles, brick (at the level of detail needed for fabrication), based on forms of aggregation
- embedded relations such as for connections and embedded elements
- non-overlapping but tightly packed relations between objects, such as precast concrete pieces and slab assemblies
- relations between objects to reflect different semantics: connection, association, assembly
- alternative model views for the same object, for fabrication, as installed (deformations), and analytic models
- and others.

These issues require full understanding by the relevant users, and their unambiguous mapping to IFC for intelligent exchange. The level of detail in the provided and exchanged models for each information unit can vary based on the project stage, purpose of model exchange, model recipient, and local practices. Further, different delivery methods impose changes in roles and responsibilities of project parties, which considerably change project deliverables at each stage for each discipline involved in the project. MVDs need to satisfy particular level of detail requirement, for example, if embeds are included or not, rebar is included or not, surface finishes defined, etc., for each phase of the project. Studies on managing data exchanges, by reducing or simplifying the information, show that without well-defined model views, the current approaches are vulnerable to errors, omissions, contradictions and misrepresentations [8]. Most of these inadequacies can be related to the lack of semantic uniformity among BIM tools in mapping their internal objects to IFC entities and properties. For example, there has been no standard procedure in which a precast architectural facade is modeled and mapped to and from the IFC schema [70]. The results of the exchange scenarios of BIM applications have been shown to contain information loss or distortions [101]. Performance studies of BIM data bases, to create partial models and run queries, show a strong need for both identifying model views for specific exchanges as

well as for specifying the exchange protocols in a consistent and formal manner [96, 106]. It can be noted that, human cognition is flexible and agile, so a person can think of a rebar cage as a type and it being instanced in multiple columns at one instant, and then in the next instant, for a different function or manipulation, the same person can think of fabrication of a particular rebar shape + material + diameter combination as a type but applied in multiple different cages, for different columns and even beams. Chapter 5 explains a scenario where (See Figure 22) three different cases of reinforcing element aggregation and each case can be used as a type. Software is generally less agile, and so the typing is fixed and embedded, which means there is a need to know in advance the internal rules for typing and instancing for the two applications to interoperate.

The current status of model exchanges using IFC is summarized as follows [117, 119]:

1. The development of an Information Delivery Manual (IDM) is based on industry knowledge and human expertise.
2. The translation from IDM to MVD is manual and tends to be error prone.
3. The base Concepts are not strictly defined.
4. Not based on logic foundations, hence no possibility of applying reasoning mechanisms.
5. The required level of detail of model exchanges not specified.

The following chapter explores the different mechanisms to formally represent information from the area of computer science and knowledge representation and sheds light on the suitability of such approaches to the issues faced in AEC-FM.

CHAPTER III

FORMAL METHODS FOR KNOWLEDGE REPRESENTATION

This chapter provides a literature review on knowledge representation systems followed by comparison of research in other areas such as semantic web to the problems addressed in this research in the AEC domain.

3.1 Knowledge Representation Systems

Formal specification of languages for improving machine readability evolved as a result of research in the field of artificial intelligence.

Knowledge Representation: Knowledge representation is the field of artificial intelligence that focuses on the design of formalisms that are both epistemologically and computationally adequate for expressing knowledge about a particular domain [5].

Description logics (DL) and *Frame-based systems* were the tools that enabled the researchers in this field to represent knowledge in a structured and reasonable way. A DL model consists of a domain and an interpretation function. The domain is the set of objects and the interpretation function is a mapping from individual, class, and property names to elements of the domain, subsets of the domain and binary relations on the domain. The key feature of DLs is that they are formal languages with well-defined semantics. Importance is given to certain formally defined relationships (subsumption, equivalence etc) between the objects. Description logics help information defined by ontologies to be shared without any misinterpretation of the terminology and their meaning. The need for this formality was observed in the many number of ways model views can be generated using IFC for the same exchange [33]. Further, it is possible to check consistency of classes and model view definitions by automated reasoning of business rules when terminology is formalized. The

Table 3: A comparison of Object Oriented Programming systems with Frame-based and DL-based systems [76].

OOP systems	**Frame systems**	**DL systems**
Instance	Frame, instance, individual	Instance, individual
Attribute, instance variable	Slot	Role, attribute
Value	Filler	Filler
Class, type	Frame, schema	Class, concept

DL Handbook provides considerable insights about the trade-off between providing expressivity and tractability. It is a computational balance, which is important in the case of rich schemas like IFC. It is important that the constructors and axioms developed assume practical decision procedures. Boolean operators such as intersection, union, and complement are widely used for geometry. Property hierarchy is an important aspect of IFC, which provides transitive properties as well as restrictions on properties. Inverse properties are widely used in IFC, which also provides enumerations for classes.

Frames paradigm can be seen as an alternative to Logics with the goal of making ontologies easier to read and understand. Frame based languages represent each class as a frame, where information about the hierarchical structure of the class, name and other properties (slots) etc are defined. A class frame is semantically equivalent to a Description Logic axiom. An advantage in frames is evident when users with different expertise work together to develop an MVD. A comparison of terminologies or features of object oriented systems, frame based systems, and description logic based systems are given in Table 3. In object-oriented paradigm a we define classes, where as in frames paradigm a frame represents a class. Instances of classes in OOP systems are called individuals in frames and DL systems. Similarly slots in frames represent attributes of a class. Thus OOP systems, frames and DL systems are different but comparable systems.

3.2 Engineering Ontologies

The idea of a formal language for knowledge representation can be traced back to the implementation of KL-ONE for structured inheritance network. Formal ways to deal with

ideas such as description, attribute, concept, role, inheritance and instantiation were introduced which paved the way for future ontology languages [21]. Some of the language formats developed over the past two decades are reviewed here, with the idea of representing EXPRESS/IFC based data schema.

3.2.1 Introduction and Overview

Ontology: An Ontology is a formal specification of a shared conceptualization [52, 19, 54].

There are several classifications of Computer Sciences ontologies, based on different parameters. Guarino [53] classifies them by their level of generality as:

- top-level ontologies, which describe domain-independent concepts such as space, time, etc., and which are independent of specific problems;
- domain and task ontologies which describe, respectively, the vocabulary related to a generic domain and a generic task;
- and, finally, application ontologies, which describe concepts depending on a particular domain and task.

Van Heijst, Schereiber and Wieringa [116], classify them according to their use as:

- terminological ontologies, which specify terms used to represent the knowledge;
- information ontologies, which specify storage structure data; and
- knowledge modeling ontologies, which specify the conceptualization of knowledge.

Fensel [41], classifies ontologies as:

- domain ontologies, which capture the knowledge valid for a particular domain;
- metadata ontologies, which provide a vocabulary for describing the content of on-line information sources;
- generic or common sense ontologies, which capture general knowledge about the world providing basic notions and concepts for things like time, space, state, event, etc;

- representational ontologies, that define the basic concepts for the representation of knowledge; and
- method and particular tasks ontologies, which provide terms specific for particular tasks and methods. They provide a reasoning point of view on domain knowledge.

Gomez-Perez, Fernandez-Lopez and Corcho [49] classify ontology based on the level of specification of relationships among the terms gathered in the ontology, in:

- Lightweight ontologies, which include concepts, concept taxonomies, relationships between concepts and properties that describe concepts.
- Heavyweight ontologies which add axioms and constraints to lightweight ontologies. Those axioms and constraints clarify the intended meaning of the terms involved in the ontology.

Some of the required criteria for ontology developed for IFC are as follows.

Syntactic and Semantic: Should have an easy to read syntax and at the same time be semantically precise.The scope of IFC covers the entire AEC industry and the information to be represented is rich and redundant. There is a need for a formal and strict syntax, however, at the same time there is a trade-off between expressivity and reasoning. XML schema is very popular in the EXPRESS community as the mode of information exchange. But it lacks the reasoning abilities of an ontology language. More recently NBIMS has developed a methodology on how to proceed from a requirements specification to model view development. Most of the information is depicted in natural language or UML. But there is a need for a syntax, which makes it easy to build a knowledge base and at the same time satisfies the needs for reasoning.

Expressive Power: IFC requires information to be associated with classes and properties in a hierarchical manner, which cannot be provided by Description Logic alone. At the same time the local typing for property values and a rich set of data types is also required.

Computational and Reasoning: Care needs to be given not to develop formalism for IFC, which makes inferences un-decidable.

All these requirements for a formal representation scheme for IFC can be summarized as below:

- Finite controlled (extensible) vocabulary
- Unambiguous interpretation of classes and term relationships
- Strict hierarchical subclass relationships between classes
- Property specification on a per-class basis
- Individual inclusion in the ontology
- Value restriction specification on a per-class basis
- Specification of disjoint classes, inverse relationships, part-whole relationships
- Specification of arbitrary logical relationships between terms

3.2.2 Ontology Languages and Editors

Knowledge Interchange Format (KIF): The knowledge Interchange Format is a formal way for knowledge exchange among computer programs that are different in nature [46]. The semantics of KIF are based on the correlation between the terms and sentences of the language and the conceptualization of the world in terms of objects, functions and relations. KIF uses declarative semantics for representing the meaning of expressions using first order predicate calculus and reasoning rules. This is a very early approach and lacks in its inability to transmit declarative information between large systems. Moreover, a declarative database created for IFC should be more than just a collection of first-order sentences. KIF has been the foundation for much recent work like Ontolingua Server [40], which allows users to develop high level ontologies by using predefined ontological descriptions.

F-Logic: F-Logic is the marriage of well-defined semantics of logics with frame-based languages to provide formal semantics and resolution-based proof procedure. This was developed keeping in mind the demands of a database logic language comprising the object oriented features such as object identity, complex objects, inheritance, methods and rules

[72]. Inference engines for F-Logic are efficient and recent approaches to represent RDFS and OWL in F-Logic has been successful [2].

XML: Extended mark-up language (XML) was developed to separate the markup of web content from presentation, thereby streamlining task specific and domain specific data on the web. Again, this is not a semantic language and relies on Document Type Definition (DTD) to enforce constraints on which tags to use and how to use them. XML is widely popular over the web and has been successfully implemented as a exchange format for EXPRESS data, however it lacks in formulating axioms and cannot be considered as an ontology.

RDF: Resource Description Framework is the standard developed for describing any web resource in a machine understandable way while exchanging information. RDF strives to add a formal definition to the web by providing a data model and syntax conventions based on the object-attribute-value, commonly known as the RDF triplets. The main advantage of RDF over XML is that RDF can define basic vocabulary in the form of a schema or layer on top (RDFS).

SHOE: This was an approach to embed ontological information in simple HTML for web pages [83]. This approach extends HTML with a set of object-oriented tags, thereby relating the context to ontologies about the domain.

UML: Attempts were made to utilize the large user base for Unified Modeling Language and developed ontologies based on UML. However, a major drawback for UML to develop ontology is that it lacks the formal specifications that form the backbone of ontologies. UML is attractive in the graphical expressive power of a language, but as mentioned, lacks in the reasoning capacity.

DAML+OIL: DARPA Agent Markup Language and Ontology Inference Layer were unified to form the DAML+OIL, which is built on the RDF and RDFS layers with adherence to description logics. OIL is combination of frame-based systems, description logics and web standards [41], where the idea of class and its super-classes and attributes are used with relations being defined as independent entities. Concepts and roles based on description logic are used to express anything in a mathematic way enabling reasoning. DAML mainly

consists of the ontology language and a language for expressing constraints and adding reference rules.

OWL/OWL 2.0: OWL2 Web Ontology Language is the formal ontology language developed for the Semantic Web. OWL2 provide classes, properties, individuals, and data values and are exchanged as RDF documents. OWL2 data types are based on the XML Schema Definition Language (XSD). It is an extension of the OWL Web Ontology Language developed by the W3C. Main considerations for OWL2 are the syntax and semantics. This, in turn, provides the user with a variety of options.. In terms of syntax there is RDF/XML, which provides interoperability between all conformant OWL2 software. OWL/XML, providing ease of processing using XML tools, is an XML serialization. For ease of readability there is a Manchester Syntax [63] developed especially to read/write DL Ontologies,. There are two alternative ways to provide semantics, Direct Semantics or RDF-based Semantics, which is necessary for reasoning tools to check consistency, subsumption relationships and querying.OWL2 is backward compatible with OWL and ontologies created in OWL are valid in OWL2 as well. The main extensions are in the form of keys, property chains, richer data types and ranges, asymmetric, reflexive, and disjoint properties, and enhanced annotation capabilities. OWL2 also introduces the new Manchester Syntax. OWL is weak in modules and imports, has to import entire ontologies. Further, closed world assumption: a statement is true when its negation cannot be proven.

Unique names assumption: if two classes, or properties have different names, we may still derive by inference that they are same. It also does not provide property chaining and rules etc. Building ontologies by implementing them directly in ontology languages can be time consuming. The major steps involved in the development can be conceptualization, implementation, consistency checking, and documentation. A brief summary of the tools currently available for ontology development are explored here keeping in mind the overall goal of suitability for IFC.

These can be broadly classified into two categories;

- specific ontology language based tools, and

- integrated tool suites that are language independent.

The first three listed in Table 4 are language dependent and the rest are independent. The latter provides more functionality and flexibility in terms of ontology development and hence is the focus of this study.

The Ontolingua Server: This is one of the first ontology tools to be developed and it focused on the development of Ontolingua ontologies using an editor [40]. It also features a Webster, an equation solver, an OKBC server and an ontology merge tool Chimaera. Developed by the Knowledge Systems Laboratory (KSL) at Stanford University, it also provides an online repository of ontologies. The KR system is implemented in Lisp and stored as Lisp text file.

WebOnto: This uses Java applets instead of HTML forms, in comparison to Ontolingua, hence it provides better graphical capabilities for working with. Uses OCML for reasoning.

OilEd: OilEd was developed to work with OIL, DAML+OIL and OWL. It allows representing classes organized in class taxonomies, properties, property restrictions, and individuals [9]. Disjoint and exhaustive subclass partitions represented as axioms.

Protege: The concept behind Protege defined by its authors, states knowledge acquisition proceeds in well-defined stages and that knowledge acquired in one stage could be used to generate and customize knowledge-acquisition tools for subsequent stages [90]. Evolution of protege is explained in [47]. Some of the main features of protege are;

- Based on frames/owl and first order logic,
- Modeling components classes, slots, facets and instances, and
- Classes can be concrete or abstract.

Miscellaneous: Literature provides information about many other editors and tool kits, such as, WebOde [4], OntoEdit [114], KAON [84], etc. listed in Table 4.

3.2.3 Knowledge Sharing Paradigms

Structure-Function-Behavior: Structure - Function - Behavior (SFB or SBF or FBS) is a knowledge representation paradigm where information is grouped according to the

Table 4: A Comparison of Ontology development tools and salient features.

System Name	**Onto Lingua**	**Web Onto**	**OilEd**	**Protege**	**Web Onto**	**Onto Edit**	**KAON**
Developer	Stanford	KMi	Man-chester	Stanford	Madrid	Onto-prise	Karls-ruhe
Formalism	Frames FOL	Frames	DL	Frames FOL	Frames FOL	Frames FOL	Frames
Language	LOOM	OCML	DAML	*	*	*	*
Inference Engine	JTP	OCML	FaCT RACER DIG	PAL FaCT ProLog	ProLog	?	?

* - Language Independent. ? - Information not available

structure of the data, its function and behavior in order to construct a clear, consistent, computable, and widely useful model [115]. This is also known as the *Teleological Modeling* of a system, which is a representation that specifies both functions of the system and the causal processes that result in the system functions, at multiple abstraction levels. Computational research on engineering design and problem solving using teleological modeling is reported in [108, 27, 26].

At the highest level, an SBF specification looks like the following [48]:

SBFModel := [STRING // Model name, STRING // Description, *StructureModel*, *FunctionModel*, *BehaviorModel*, *Stimulus+*]

In [48] a *StructureModel* consists of the elements and the connections among the elements. A *BehaviorModel* is nothing but a set of States and related Transitions between them and the *FunctionModel* describes the purpose or goal of the element, which is achieved through its behavior. The external effects on the behavior of model is represented through *Stimuli*. SBF models of complex systems enable computer programs to draw inferences about the systems. SBF provides benefits to knowledge representation by means of precise vocabulary, consistent usage, and tool building methods for model checking, model simulation, and interactive guides and critics for model construction [48].

Semantic Web: Semantic Web is considered as the architecture of web that provides automated reasoning based on content producers ability to share ontological information to

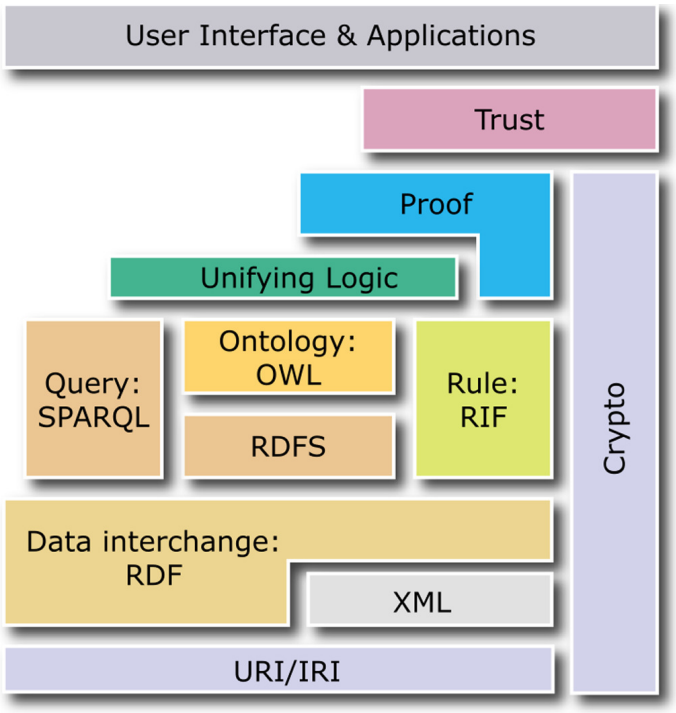

Figure 11: The Semantic Web layer cake illustrating the information representation hierarchy [12].

define formal semantics. Similar sets of issues were faced by the semantic web development effort as compared to IFC interoperability. In order to enable automated processing of data, the semantic web is envisioned as a web of linked data available in a standard format, reachable and manageable by semantic web tools. Figure 11 shows the layer of technology or information representation over the semantic web [12]. The lowest layer consists of Uniform Resource Identifiers (URI) that identify resources on the web. XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents. RDF is a data model for objects and relations between them, and provides a simple semantics using RDF schema, which is a vocabulary for describing properties and classes. RDFS provides simple reasoning techniques based on inheritance and instantiation. OWL is the semantic web ontology language that provides additional formalism on top of RDFS and supports richer reasoning and tractability. Rules, Proof, and Trust layers are what the name suggests, on top of which interfaces work. Crypto provides standards for encryption and protect the data. Semantic web technologies is being tried in various indus-

tries such as education, energy, e-commerce, health care, social networks, etc. for improved search, content discovery/management, data integration, semantic annotation, etc. Some of the key findings are listed here. Ontologies play a key role in making the web machine readable or automated by providing structured vocabularies. These define the relationships between different terms allowing applications to interpret their meaning in a flexible and unambiguous manner.

The requirements for a web ontology language are similar to the requirements identified for a ontology language for IFC, for example;

- compact syntax,
- highly intuitive to humans,
- well-defined formal semantics,
- able to represent human knowledge, and
- include reasoning properties.

3.3 Summary: Knowledge Sharing in AEC-FM Industry

The scope and potential of BIM is ever-increasing as a result of new and IT-enabled approaches to facilitate design integrity, virtual prototyping, simulations, distributed access, retrieval, and maintenance of project data between multiple disciplines [42, 111]. Integrated Design and Delivery Solutions (IDDS) [98] recognizes the need for a holistic approach to research and development to bridge the gap between collaborative processes, workforce skills, integrated information, and knowledge management. The recent reliance of 2D drawings for transfer of information is identified as a significant road block towards widespread adoption of technologies in construction [104].

The issue of how to share knowledge (exchange of model data in case of BIM) and infer meaningful information at the receiving end is an important aspect in any industry. Similar issues have been addressed in the areas of databases research, where integration of schemas is an important aspect, due to the diversity of data models and database management systems [109, 91, 39]. In AEC/FM, the previous work has identified the importance of seamless data

exchange capabilities, and the need for model views to achieve this [60, 36, 98]. Attempts were made to simplify geometry alone in a data exchange for energy analysis purposes, using a simplification tool [8, 85, 7]. Adachi [1] uses XML in a web-based architecture to communicate between an IFC model server and client machines. Other research has compared the use of IFC XML for file transfers [100, 74] and found that they cause inconvenience due to much larger file sizes than normal IFC files [96], affecting the performance of querying model data. Kiviniemi [73] describes a solution in the form of identifying the requirements and creating a requirements model to be linked with product models and asserts the importance of semantic links between various domain-specific models.

There are parallel approaches to introduce semantics into building information modeling, by means of using web standard technologies (W3C) and techniques [123, 10, 15, 11]. They include the use of formal methods such as RDF schemas, functional specifications, and ontologies to some extent. Garrett et al. [45, 112, 79] utilizes text analysis algorithms to solve classification issues, where as semi-automated approaches to integration of project documents to model objects are available [23], but the need for advancing formal methods for data collection, data mining, data analysis, and project planning and control are documented [24]. Research at Massachusetts Institute of Technology (MIT) is focusing on creating such collaboration forums using intelligent systems and agent technologies to facilitate interoperability among team members using different applications, operating systems, and platforms [102]. Lee [32, 77, 78] and Borrmann [17, 18] are working on rule-based approaches and query languages for building information models. The future research is aimed at integrating different aspects of model checking, such as clash detection, code checking, work process checking, model comparison, etc. under one single umbrella of semantics based model checking. Such approaches follow a semi-automatic mechanism of pre-defining some of the semantic based concepts using a mark-up methodology and development of computable rules [62, 61]. Another recent effort uses a combination of a neutral schema; called GMSD [121] along with an XML based extraction approach to filter IFC and non-IFC data [71]. A set of varied tools has been developed to perform filtering at class level and object level to create multi-modal views by linking attributes at the top level of XML

schema. However, the use of different tools and generation of multi-model views gives rise to issues such as i) tackling of multiple views within the same application, ii) inter-linking these multiple views to an interoperable system, and iii) maintaining these links for future reuse. Moreover, the scalability issues of XML schema are still unresolved. Interlinking of information across process and object models by structuring information in coherent model hierarchies are also examined [107]. However, at this point of time, many of these approaches are limited to custom domains and specific applications and are not applicable as a general and industry wide solution. IFC Solutions Factory [13] is a shared repository for model view development work around the world. Most of these efforts are generating concepts with IFC bindings in an ad-hoc manner suitable to their specific application requirement, with limited re-use across such efforts. The above discussion illustrates the pressing need for a more formal approach at developing model views and also making the basic concepts (SEMs, in this research) consistent across domains.

CHAPTER IV

RESEARCH HYPOTHESIS AND METHODOLOGY

This chapter explains the research methodology for formalizing the exchange modules using an ontological backbone and mapping them to IFC entities, relations, attributes, and functions. The first section explains the research hypothesis, followed by an overview of the framework. The subsequent sections deal with different phases of the research, such as the modeling criteria and assumptions considered in the research and then walks through the different steps of defining a ontological system. This is followed by the Semantic Exchange Module generation, to be used in the development of model views. Finally, detailed plans for validation and verification of the research are addressed.

4.1 Introduction

The overall research methodology can be envisioned as involving the development of four different but overlapping models. These are as follows:

1. Requirements Model
2. Conceptual Model
3. Implementation Model
4. Validation Model

The term *Model* in this regard does not refer to a building model created from a software application, but rather the notion of a scientific model as given in [20]. The partitioning of the research into four such paradigms is graphically shown in Figure 12. The requirements model is the output obtained from analyzing the IFC schema for embedding semantics such that information exchanges are possible. The conceptual model is comprised of the ontological definitions, which are abstract in nature (high-level). The implementation model

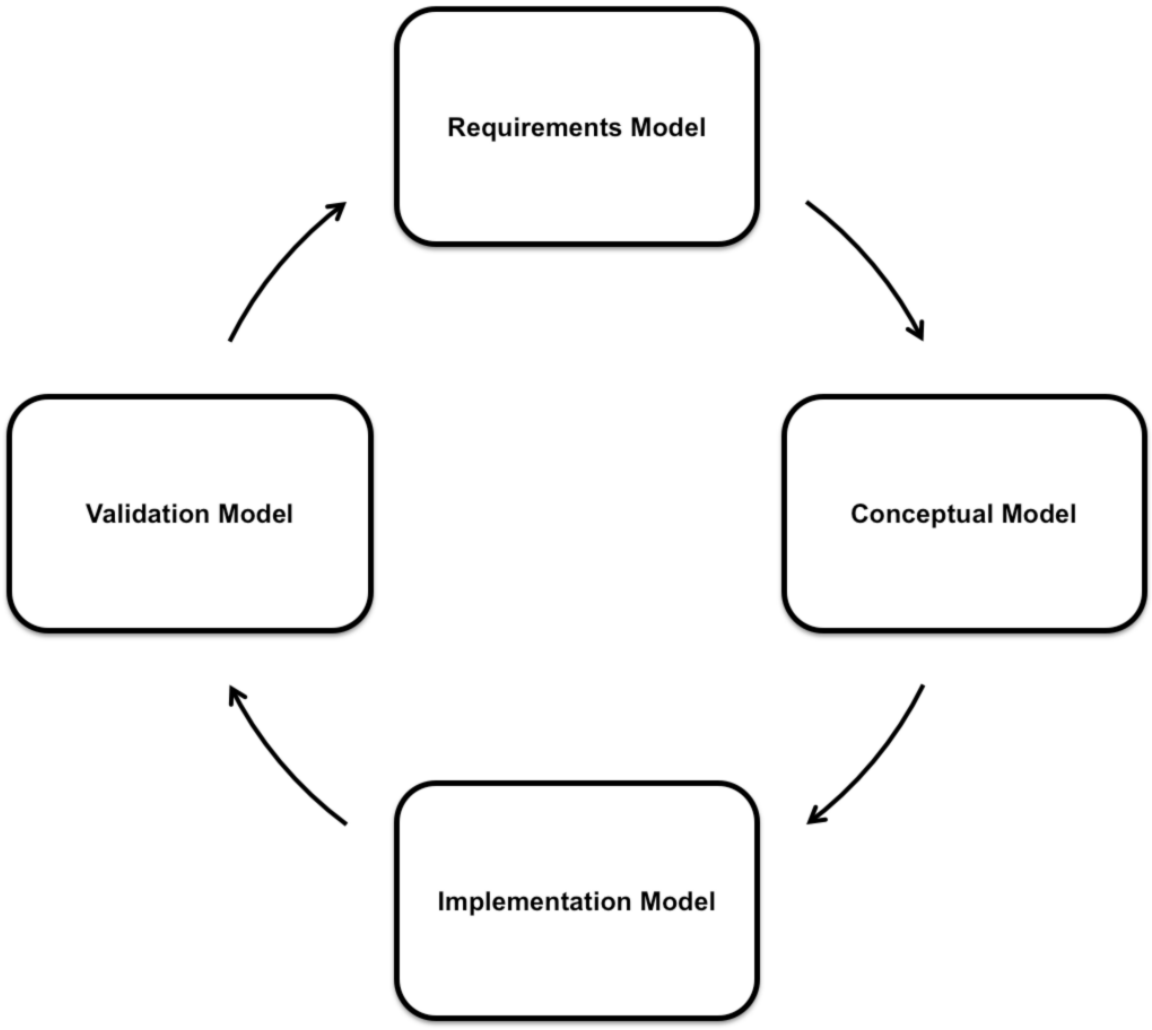

Figure 12: Model cycle: Compartmentalizing the research into four paradigms.

is the result of mapping the conceptual model to the IFC schema for implementation of Semantic Exchange Modules. The validation model comprises of the metrics for validating and verifying the research to enable it to be broadened and applied to several domains.

4.2 Hypothesis

Formal specification of Semantic Exchange Modules using engineering ontologies can make the precast building model exchanges consistent, modular, testable, and reusable. Two important research questions were put forth during the development effort of the Precast NBIMS that formed the basis for initiating this research and are central to answering this hypothesis. These are as follows:

Research Questions:

1. Can a well defined method for semantic classification and representation of IFC entities be developed? and,
2. How to develop model views that are consistent, testable, and reusable across various domains in AEC/FM industry?

4.3 Overview of the Framework

This research is divided into four phases. The first phase involves the analysis of IFC product model schema to identify the suitability of embedding semantic meaning and related issues in model exchanges. The second phase involves development of an ontology-based definition for IFC based on the analysis from the first phase. This phase also involves the preparation of a knowledge base, setting up the modeling criteria and assumptions etc. It is envisioned that domain knowledge will be divided into smaller pieces called engineering ontologies, such as geometry, type-instances, assemblies, connections, properties, etc. and the application ontology can be constructed by the integrations of these smaller ontologies. The third phase involves the creation of SEMs as a mapping of the ontology to the IFC schema. The final phase of the research involves realizing the validation and verification of the approach so that it can be used in various applications. Some of the applications identified include, but are not limited to, developing modular MVDs, satisfying model exchanges, querying etc. As a proof of concept, guidelines for a new model view development methodology and model view developer plugin are provided, based on SEMs. Further, enabling should also involve the evaluation or impact assessment of the methodology. Figure 13 shows the high level overview of the research methodology, and it is explained in detail in the following sections.

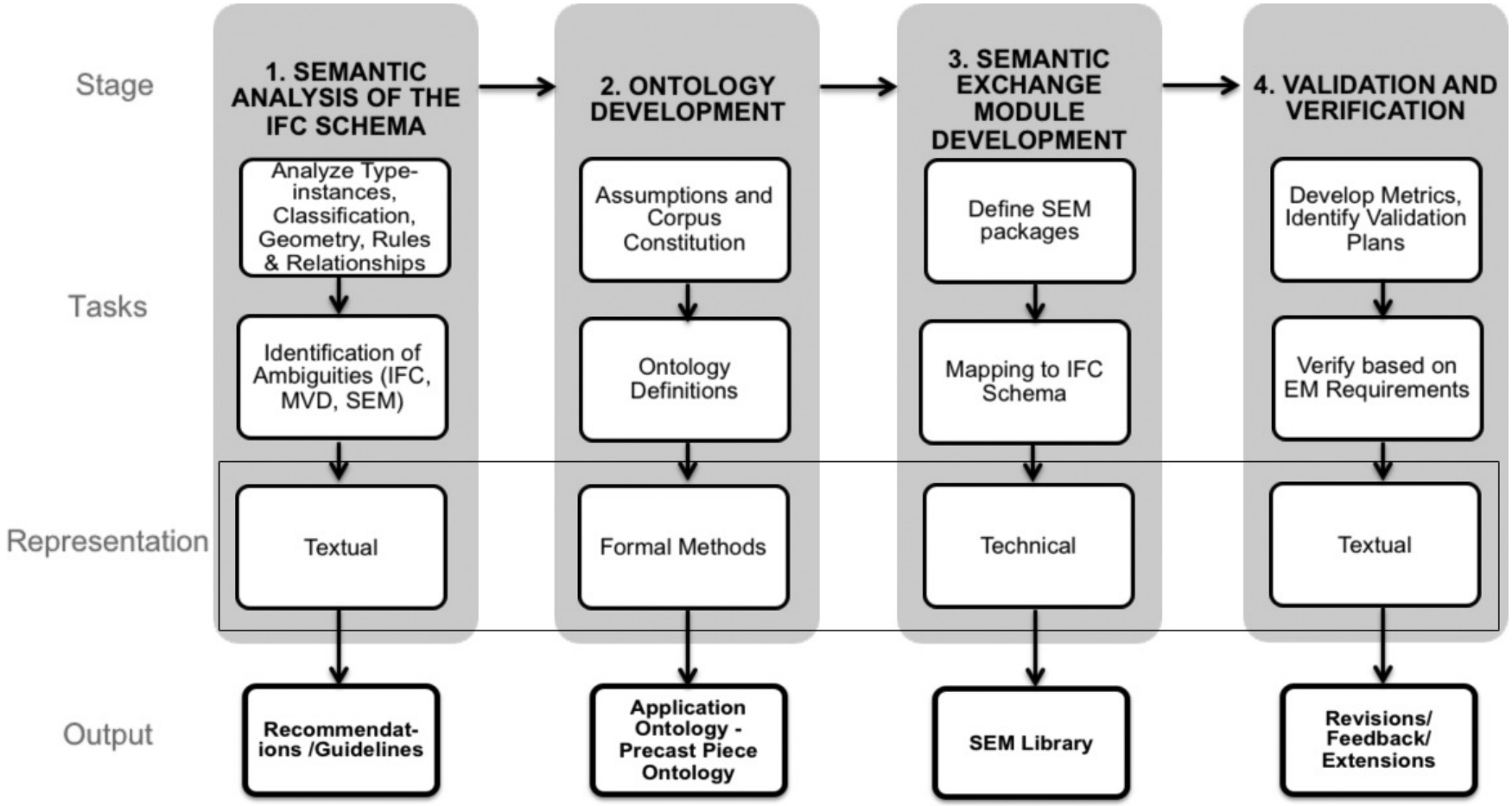

Figure 13: Main steps in the research methodology.

4.4 Semantic Analysis of IFC

The first step in the methodology is an in depth analysis of the IFC product modeling schema. Such an exercise forms the backbone of the ontological definitions to be developed and provides the *Requirements Model*. IFC lacks the formal semantics to relate its entities and relationships to reasoning mechanisms. There is no firm class or property hierarchy as most classes and properties are direct subclasses of the top level concept . Further, several relationships take *IfcRoot* as their domain or range. Some of the implicit semantics are in the IFC documentation whereas some semantics are explicit and left to the users assumptions (for example, geometry exchange) or to be defined by future work (for example, IFC specifications for construction safety). To overcome this situation, the level of commitment in IFC will need to be raised if it is to support the tasks such as modular testing, querying, and transaction based services. Figure 14 illustrates the notion of formalizing the IFC schema. The output of this stage are the recommendations and guidelines to improve the ambiguities of IFC schema in textual format as shown in Figure 13. The detailed analysis of IFC schema, the results, and recommendations, constitute the Chapter 5 of this thesis.

4.5 Ontology Development

The results provided by the semantic analysis are relatively informal. The development of ontologies is a key aspect in knowledge representation and sharing, as it provides the bridge between human and computer understanding. The ontological definitions are developed by adding detail where necessary, incorporating formalisms wherever helpful, and generally enhancing the consistency and modularization of IFC entities, attributes, and relations. The result of this phase is called the *Conceptual Model*.

Conceptual Model: The Conceptual model is defined as the consolidation of all relevant information and features of the system under investigation in a format that is as concise and precise as possible. It serves as the mapping between the requirements and the implementation of Semantic Exchange Modules.

The formalism is achieved by working from a high level of abstraction to a lower level of

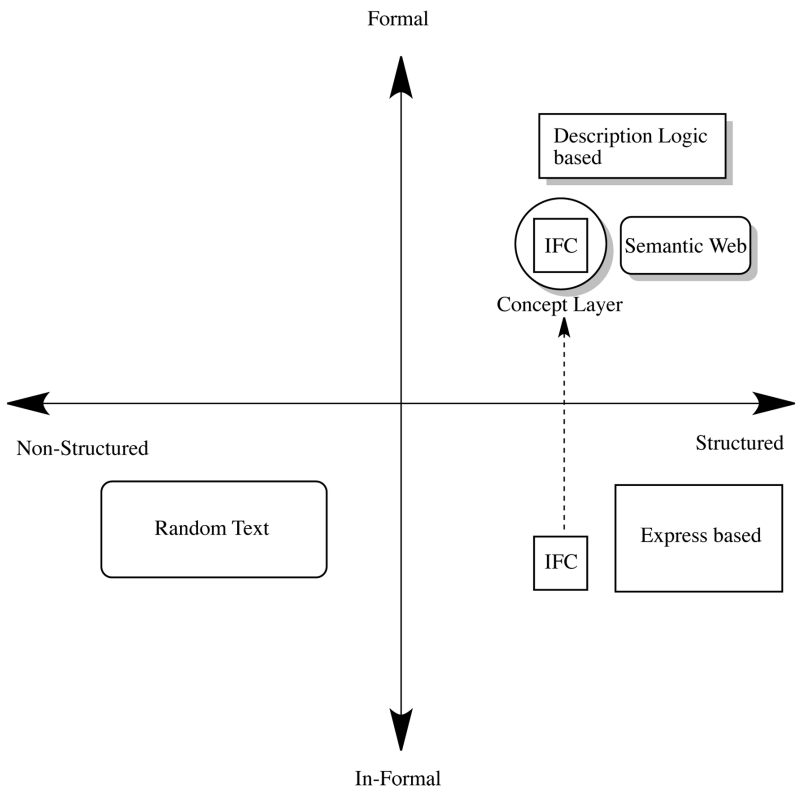

Figure 14: The objective of the research to improve formalism of IFC.

application. The objective of ontology development is to remove the ambiguities associated with differing viewpoints as shown in Figure 15. To illustrate this, let us consider a *usecase* involving information exchange between precaster and the structural engineer. A precast piece such as a floor slab can be represented as a single slab entity in the *structural model* developed by engineer, whereas the *fabrication model* developed by the precaster will include the high level description of precast piece detailing (multiple double tee slabs), connection details, finishes, joints, embeds, reinforcing, tensioning cable layout and block-outs, pretensioned pieces, and lifting hooks, etc. Moreover, the corresponding IFC entities, relationships, attributes, and properties will also be different. For example, Figure 16 shows a process map with the particular usecase highlighted. Ontologies provide a means to remove ambiguity in such scenarios, by representing knowledge using formal methods. In this research, OWL language [64] and Protege editor [47] are used as the tools for ontology development. The various steps involved in developing an ontology and results are explained in Chapter 6 of this thesis.

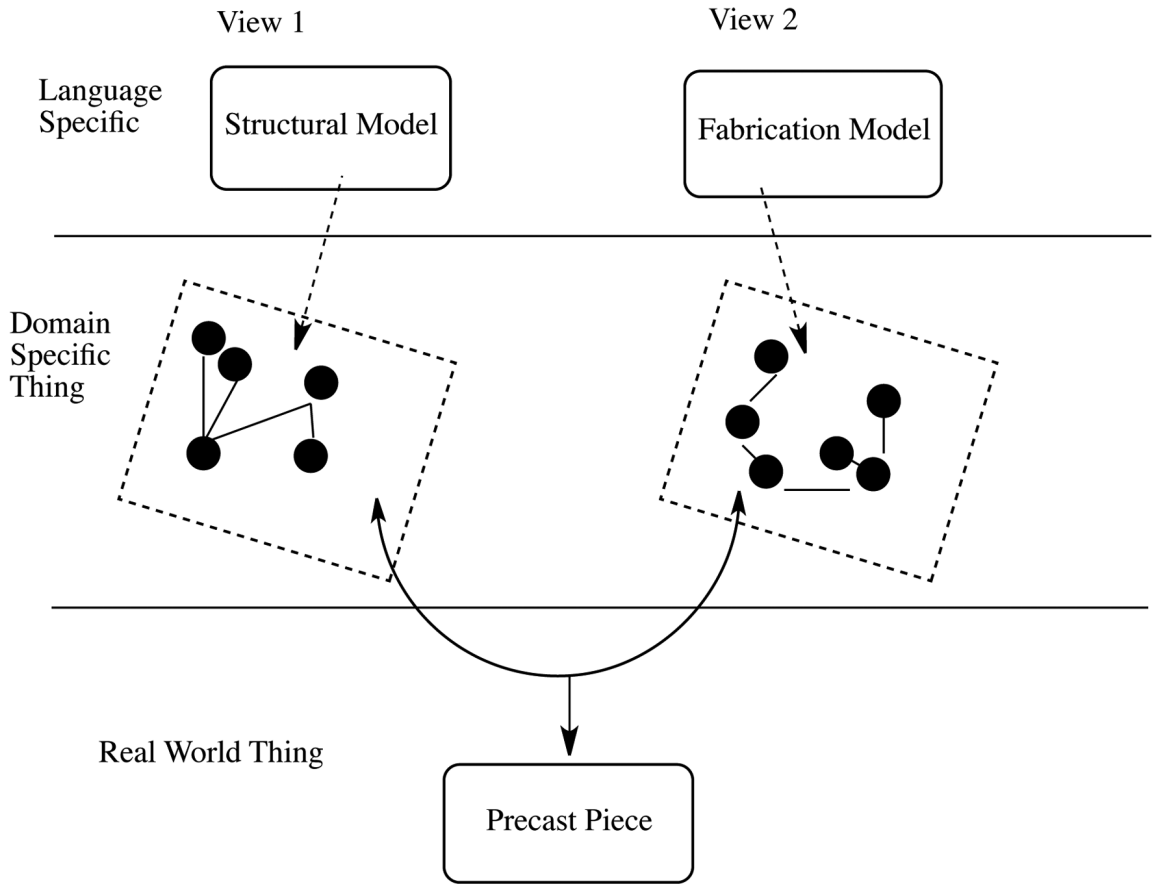

Figure 15: The different viewpoints of objects in different domain languages.

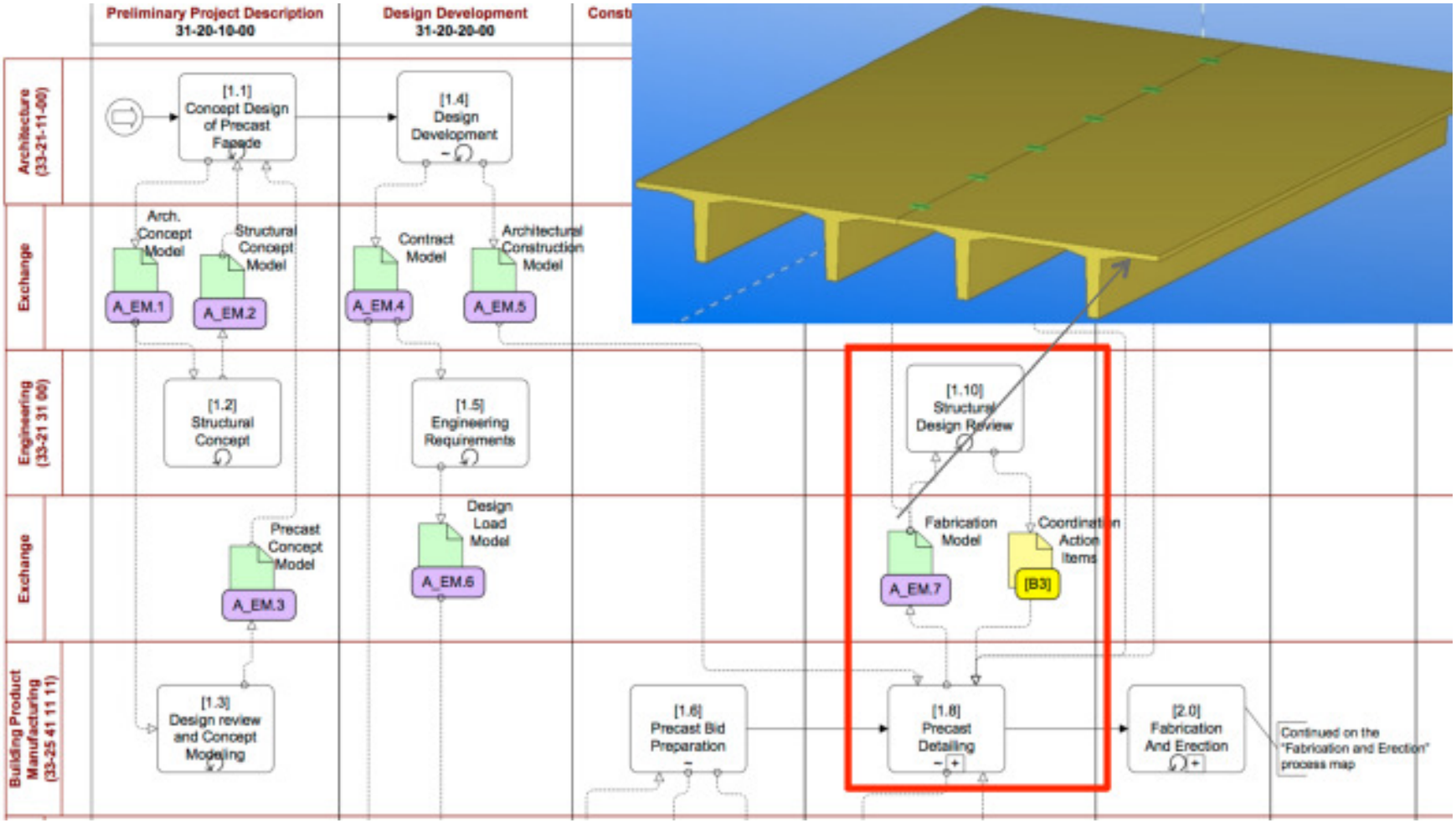

Figure 16: Model exchange between a precaster and engineer - *Fabrication Model* with example Double Tee [34].

4.6 Semantic Exchange Module Development

The precast piece application ontology is developed using Protege and is available in OWL form. However, there is limited use for ontology in the form of OWL representation, and an additional layer of mapping is provided for implementation. Semantic Exchange Modules (SEM) provide this additional layer. IFC entities, relations, attributes, etc. are mapped onto the application ontology developed in the previous phase. The output of this phase is a set of semantic exchange modules, collected in the form of a library. Moreover, graphical representation of SEMs is much more intuitive for domain experts and practitioners to work with. SEMs are defined as intermediate to natural language and computer programming languages. This allows ontology language representations such as OWL and modeling representations such as IFC to stay invisible to the end user (similar to data hiding in object oriented design). This significantly lowers the barriers for practitioners (software developers). Refer to Chapter 7 of this thesis for detailed steps leading to the development of SEMs.

4.7 Validation and Verification

This is the model validation and verification phase of the project. Validation and verification are concerned with ensuring credibility and reliability of the products developed. In software engineering parlance, the validation and verification can be defined as answering the following two questions [14]:

Validation and Verification:

1. Verification: Are we building the product right?
2. Validation: Are we building the right product?

For the purpose of this research, these questions can be translated as, verification focuses on verifying the correctness of the Ontology and SEM structure developed, and validation equates to checking whether the SEMs developed actually improve the overall interoperability of BIM tools. Chapter 8 of this thesis explains the suite of metrics proposed to

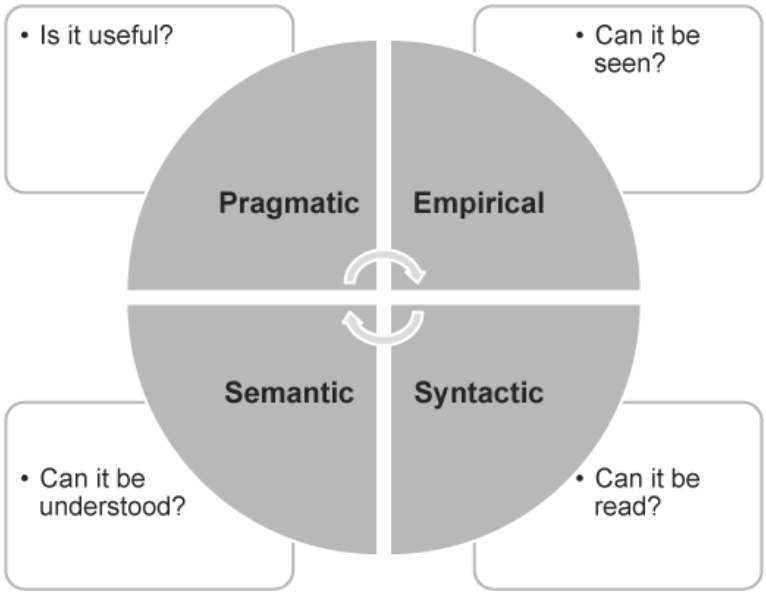

Figure 17: Illustration of the four components of the research to be verified.

verify this research. Four components of the research are verified as shown in Figure 17. Reasoning engines and test probes are used to verify the ontology structure (Semantic and Syntactic verification). Graphical representation schemes and visualizations provide empirical verification of the research. A model view for precast is compiled based on SEMs and tested for correctness using parsers for EXPRESS schema (Syntactic).

Validation testing of model exchanges can be broken into four levels:

1. Checking the syntax and structure of project exchange files for conformance to the IFC standard (IFC 2x3, or 2x4 etc.) this validation only applies to the export functionality of any given BIM software tool. It is not useful to test import routines this way, as import does not generate data that can be externally tested.

2. Checking the objects in a project exchange file, as well as their properties and relationships for conformance to the bindings stipulated for them in the relevant MVD document. This test validates that the tested application can generate an exchange file with the required objects, and that these satisfy the rules of the bindings in terms of relations and attributes. The bindings for a set of SEMs are aggregated into different ways for different MVD exchanges. Thus conformance testing is performed separately for each exchange. This too is an export functionality test.

3. Checking the import functionality of a BIM software tool for its ability to properly import the full set of SEMs defined in an MVD. This can be done using a predetermined set of IFC test files that aggregate sample instances of all the SEMs defined in the MVD. Since each possible exchange exploits a certain subset of SEMs, any given BIM software tool export function can be tested for a given exchange by testing its import of a subset of the IFC test files. This test applies to unit testing.

4. Checking the completeness of the contents of a project exchange file (objects, parameters, and their values) between two applications, to ensure that the exchange contains all of the information required for the given exchange by the definitions of the Information Delivery Manual (IDM). This check can only be performed within the context of a precast construction project, as it check content within project context. It is an export and import test.

To understand the scope and detail of the exchange capabilities needed, a test model is created. A model view developer plugin is built on top of C# libraries to illustrate the workflow of specifying and exchange models using SEMs. The plugin allows any BIM software to specify a model exchange based on SEMs, resulting in a modular MVD. These are explained in detail in Chapter 8 of this thesis. Detailed and specific test cases are developed to validate the research and explained in Chapter 8. The validation of the research is a long term goal and plans have been identified for this as well. Implementation plans for SEMs in the industry are scheduled with a set of software developers, to implement and test the idea of modularizing the structure of model views into composable units (Pragmatic). The expected benefit for a software developer is that these only have to be specified, implemented, tested, and certified once. After that, given the appropriate software environment, any domain group or project team could implement a new model view in a short time by using previously defined and implemented SEMs. This should greatly reduce the effort required by software companies to develop exchanges for additional MVDs. This is a long term effort that can only be validated over a period of time.

CHAPTER V

SEMANTIC ANALYSIS OF IFC

This chapter presents the results of an analysis of the Industry Foundation Class (IFC) data schema. The suitability of the IFC schema for embedding semantic meaning in a formal and consistent manner for model exchange purposes is emphasized. The topics are grouped based on type-instance issues, classification problems, geometry, relations and rules, etc. A brief explanation of the theory and definitions of the terms used in this research is given to provide the reader with an understanding of the computer programming language terminologies. The analysis is summarized at the end of this chapter and recommendations are grouped according to improvements for IFC schema, model view definitions, and Semantic Exchange Module development. These recommendations are used in the forthcoming chapters for developing SEM structure and model views based on SEMs.

5.1 Introduction

A *language* is usually defined as consisting of any valid sentence that can be formed according to a set of rules, which is the grammar. The rule-set can be infinite as long as the grammatical requirements are met. In other words, languages include a theoretical description of the structure, composition, and its constituents [30]. In terms of a computer programming language *the description of how the language is composed and what its constituents are* can be defined as the *syntax* and *semantics*. Both the syntax and semantics of a programming language must be specified precisely. The semantics of a programming language is the mapping defined by the compiler from the program written by a human to the target code executed by the computer. Gunter [57] explains syntax as the context-free phase of the compilation, and semantics as the phase where types are checked, code is optimized, etc. Further, to make the language more portable, if a compiler for one machine has specified the language, then either the second machine on which it needs to be implemented

should simulate the first machine or the compiler must be translated in a compatible way. For the purpose of this research, an *Object* is defined to consist of data and methods. Usually data structures and methods that operate on them are grouped together as a set of abstract data type [58]. Similarly, in the case of IFC schema, all physically tangible items, such as beam, columns, wall, spaces, processes, etc. are grouped together into an abstract supertype called an *IfcObject*. Figure 18 shows the definition of an object (with building elements being a child class) in IFC and its relationships. These relationships define the context information of the object. in The structure of a model view for the exchange of product model data between various BIM application tools depends on the extent to which building function, engineering, fabrication and production semantics will be embedded in the exchange model at the source and the capability of any receiving application to comprehend them. Classification schemes is an example of attaching these semantics in a context-free manner (explicitly).This is possible, if sending and receiving applications follow the same conventions that determine their meaning. Together, the syntax and semantics should specify a model view precisely. Syntax is easier to understand, but it is the semantics that specify the meaning or context of the information. At one end of spectrum, an exchange model can carry only the basic solid geometry and material data of the building model exchanged. The export routines at this level are simple and the exchanges are generic. In this case, for any use beyond a simple geometry clash check, importing software would need to interpret the geometry and associate the meaning using internal representations of the objects received in terms of the software's native objects. At the opposite end of the spectrum, a *semantic-rich* exchange file can be structured to represent piece-type aggregations or hierarchies. These hierarchies define design intent, procurement groupings, production methods and phasing, and other pertinent information about the building and its parts. In this case, the importing software can generate native objects in its own schema with minimum effort. Based upon predefined libraries of profiles, catalogue pieces, surface finishes, and materials the importing software does not require explicit geometry or other data in every exchange. The export routines at this level must be carefully customized for each case since the information must be structured in a manner suitable for the importing

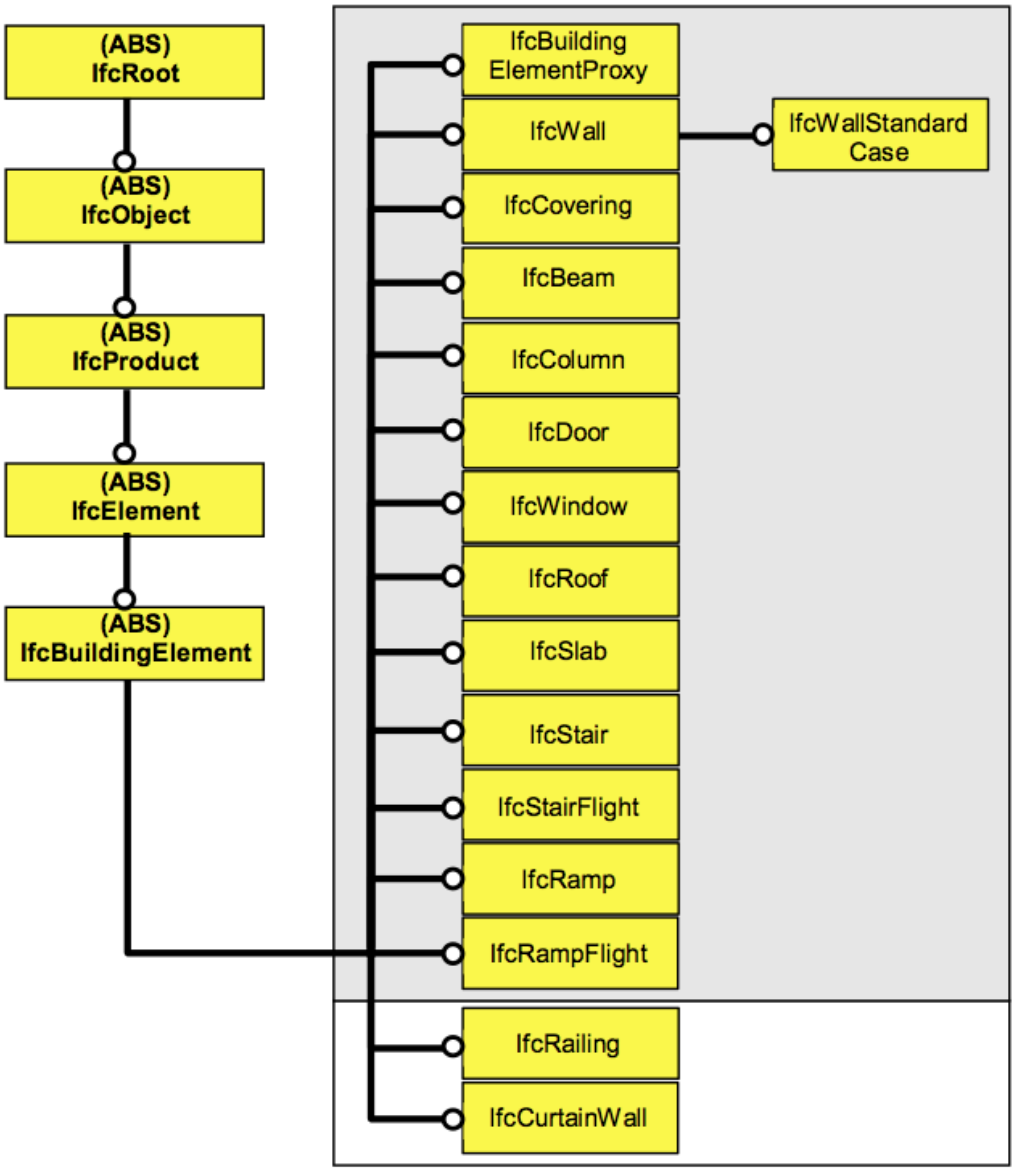

Figure 18: Definition of an Object (building element) in IFC as *IfcObject* and the corresponding relationships [81].

applications supporting each use case.

Different use cases require different information structures. For example, an architect might group a set of precast facade panels according to the patterns to be fabricated on their surfaces, manipulating the pattern as a family; an engineer might group them according to their weights and the resulting connections to the supporting structure; a fabricator might group them according to fabrication and delivery dates. In order for the importing application to infer knowledge from the exchange, the exporting application should structure the data based on the grouping scheme accepted at the receiving end. This can be achieved if applications at both ends follow an agreed upon standard format. This is an important requirement and needs to be taken into account when the model exchange requirements are specified.

In preparing a set of MVDs, information modelers must determine the appropriate level of meaning and the typing structure required by the IDM. If the structure is too simple, the exchanges will only have value for importing software that is able to apply some level of expert knowledge to interpret the information. If it is too rigid, then it will only be appropriate for a narrow range of use cases. This may lead to a need for large number of model view definitions. This would require software companies to prepare multiple export - import routines. Hence modularization of MVDs is important to allow reuse. This thesis investigates if Semantic Exchange Modules (SEM) and ontologies help in suitably achieving such a reuse.

The following sections elaborate on a number of aspects that must be considered during the decision making process of binding SEMs that require semantic content to an exchange schema. Although any exchange schema could be used, this research focuses on the IFC schema and highlights areas where special attention is needed to accurately and precisely specify semantic meaning.

5.2 Type Casting and Inheritance Structure

What is a Type? The Type of an Object determines its representation and constrains the range of abstract objects it may be used to represent [113].

Untyped refers to the existence of only one type. For example, in the case of bit strings in computer memory, which is an untyped universe, everything had to be represented as bit strings, i.e characters, numbers, pointers, structured data, programs, etc. This poses a drawback as there is no identifier to specify the data represented in a piece of raw memory, without delving into the depths. Such a system prompted users to start categorizing information in different ways for different purposes, according to usage and behavior. A simple illustration in the above case of bit strings in computer memory is that characters are distinguished from operations. Such categorizations led to the creation of well-defined typed systems [25].

Historically, there are four motives for the development and use of types in programming languages. Initially the idea of typing was explored to enhance the efficiency of programming languages such as FORTRAN. Later, their role as a programming discipline was appreciated to enforce restrictions on the shape of well-formed programs to detect compile-time errors. More recently types have been used to support data abstraction and modularity. In addition to the above motivations, types can also be used as a conceptual tool for classification, such that semantics will classify objects according to their structure and use, thereby elevating them conceptually above the syntax that they are expressed in. This abstraction is the notion of types that we are interested in for this research.

Typed systems impose constraints that help to enforce correctness by restricting the expected properties of data types and operations on data objects. This is a way of protecting the underlying un-typed concepts from arbitrary or unintended use. This is usually achieved by way of imposing a type structure. Programming languages such as FORTRAN and ALGOL 60 distinguished between integers and floating-point numbers, and contributed to the evolution of the concept of type. SIMULA, the first object-oriented language used the notion of type for classes. Subclasses inherit declared entities from super classes and may define additional operations and data that specialize the behavior of the subclass [25]. MODULA-2 made popular the idea of modularization as a way of structuring. The usefulness of a type system lies not only in the set of types that can be represented, but also in the kinds of relationships among the types that can be expressed. This provides the ability to compute

or infer knowledge from the types to validate the correctness of the desired relationship. The semantics of having a type can be interpreted as a membership in the appropriate location in a *lattice*, formed by all the legal values allowed for that particular type. The top of this lattice can be considered as the set of all values and the bottom the singleton set containing the least element. Only those subsets are *legal types* that obey certain properties known as *ideals*. The precise definition of such ideals ensures the strength of the typing system. A language such as EXPRESS specifies the ideals strongly ensuring a strongly typed system, whereas IFC, which is written in EXPRESS, leaves some of the ideals to be defined at a later stage to support multiple views of objects. This makes IFC a *weakly typed* system. Type systems can be classified on an order of generality, starting from simple first-order systems, which then branch out as higher order type systems ranging from monomorphic to polymorphic, and inheritance systems with single to multiple inheritance hierarchies.

A particular schema such as IFC does not fall on any particular level of classification. Based on the review presented above, it can be concluded that IFC is a *weak typing* system, allowing rich and multiple representations. IFC allows polymorphic representations but restricts itself to single inheritance. Figure 19 shows a classification of different computer programming languages. The x-axis represents the order of the types allowed, starting from simple first order types to higher order types. The negative y-axis represents the inheritance structure allowed from single to multiple inheritance. Multiple inheritance allows entities to inherit attributes from multiple parents. It should be noted that the IFC can be classified as a higher order type system with proposed introduction of *IfcAssemblyType* in Release 4. In Figure 20 the spectrum of possibilities involved when defining a model view in terms of exchange semantics are identified. The first dimension is the range of possibilities along the spectrum and denotes the degree of typing a model view definition can require. This is expressed by the depth or breadth of hierarchical classification and aggregation to be used. It is possible to layer a classification schema in two ways. The first method layers them strictly hierarchically, with each instance object belonging to just one type grouping, while the second method uses a distributed manner, where each instance inherits properties from multiple object types. The range varies from independent instances (on the left), through

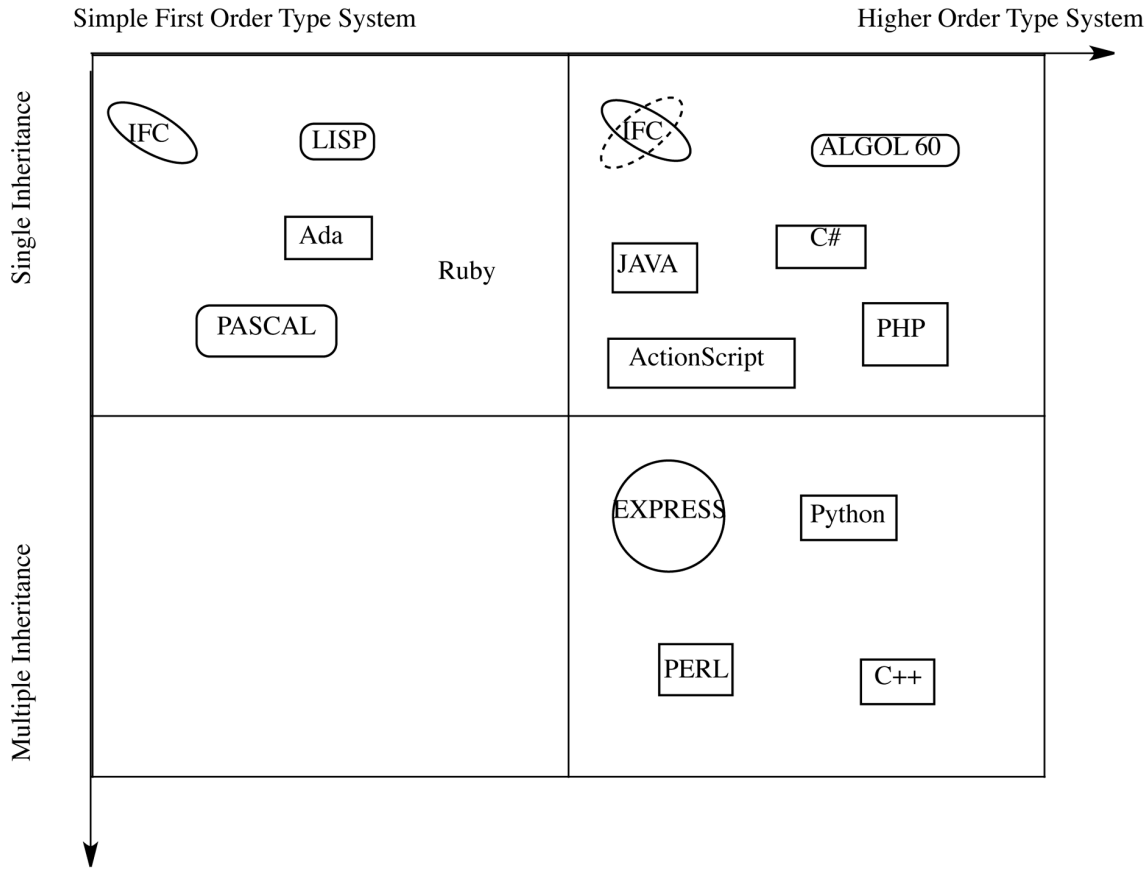

Figure 19: Classification of computer programming languages on the basis of Type and Inheritance structuring.

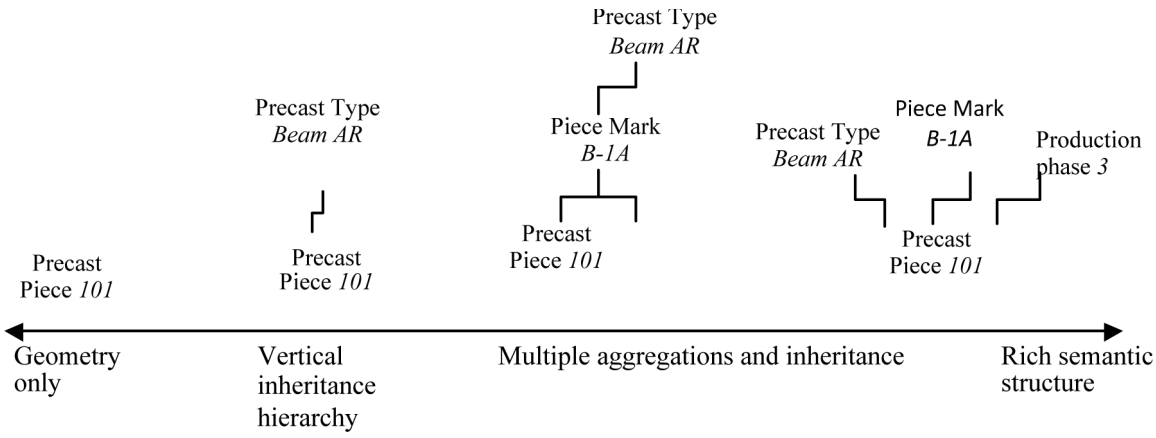

Figure 20: Spectrum of possibilities in defining model views [119].

weak typing and through deeper and stricter inheritance trees without multiple-inheritance and relationships between type and instance objects at run-time with multiple-inheritance (on the right). A detailed discussion on polymorphic type representations and different mechanisms to handle them in programming can be found in [89]. For example, consider a *piecetype*. This can be a drafting block, or turning an instance into a block (as can be done in AutoCAD) for two purposes: (1) to group it in terms of making it a type and (2) placing instances of it. In the BIM world, the issues and objectives are different. For example, the approach may require making a column, then making instances of the column, or a window style. However, just as often, we are interested in building assemblies and assemblies of assemblies, all at the type level. It should be possible to reuse these levels in other assemblies (types), and also map them to instance locations. This capability is not available in IFC (until the implementation of release 2x4, which provides *IfcElementAssemblyType*), although it is possible to design assemblies in most BIM tools in the same manner. Thus, a *type* in IFC should be an object class that can be used to define other types or instances of objects. The issue could be resolved if it were possible to obtain multiple levels of this type.

One possibility is to use a *flat* approach where all data in an exchange file is provided explicitly (i.e. an expression of the physical reality with little or no functional or behavioral interpretation). In this approach, typing is unimportant, because the receiving application or operator is assumed to be capable of interpreting the information that has been

exchanged for its own purposes. This is done manually, based on the reference data, at the receiving end, and involves a slow and tedious effort. This is evident in some of the BIM software's approach to IFC import, in which all objects in the IFC file become reference objects in the imported BIM file rather than native objects. In this approach only the human operator interprets imported data and promotes those of interest to the functional status of a native object. The typing does not matter if the receiving data is not edited. The "flat" approach seems to guide the IFC core group in hitherto refraining from providing arrays of objects [94].

Another approach is to establish a hierarchical structure that is designed to meet the needs of a particular exchange between two specific software tools. In this approach the rules are made explicit, and the typing aggregations are different according to the specific needs of each exchange. This approach also implies that multiple exchange format definitions will be required, causing the need for multiple export and import functions. The IFC object mode has the concept of object types, which is made possible through the class *IfcTypeObject* and subtypes. The most common characteristics of an object are collected together at the type level and at the instance level the object will contain the location information and occurrence specific properties and characteristics. The object type hierarchy available in IFC is illustrated in Figure 21. *IfcObject* is the super type of all abstract and physical objects in the IFC. Each Object has a set of *IfcPropertyDefinitions* that identify type-specific properties. One drawback of the hierarchical structure of IFC is that due to its size and deep inheritance structure, it is difficult to see what the definition of a single object really is. Hence, model view developers need to be well versed in the IFC object hierarchy. By providing a SEM library, which is intermediate to natural language and domain specific schemas such as IFC, the need for model view developers to be IFC experts can be addressed. They can directly understand the requirements based on the SEMs and need not be familiar with the underlying IFC entities and relationships.

The nesting of physical objects in other objects must also be considered. For example, a Precast Concrete DoubleTee (DT) contains reinforcing meshes, prestressed tendons, and also embedded steel plates for connections. A collection of DTs may also be aggregated

into a higher level object (a DT slab, which is likely to have its own shape) and have a cast-in-place concrete topping, so that the slab as a whole functions structurally as a diaphragm. In this case, we have the following entities: steel embeds, mesh, tendons, embedded into DTs, which are aggregated further into a slab. All levels of such aggregations are usually needed in different parts of the design/ construction process. Similarly, the typing-instance structure for rebar can be defined in multiple ways. If the production and assembly hierarchy is relied upon, a rebar type can be defined as an assembly structure of rebar pieces, by material diameters, strength, and coating. (Figure 22, case 1). Additionally, by using the bending patterns of ACI or other parametric codes with length, a bent piece can be defined (Figure 22, case 2). Similarly, rebars serving the same function in the same piece can be defined as a pattern by grouping them in an 'assembly in one direction'; these may be iterated to and aggregated in turn to collect multiple patterns into a cage, producing a 'rebar cage' (Figure 22, case 3). A cage can then be assigned to an instance of a piece, a column or beam mark. At present, the issue of patterns or arrays of multiple instances of the same entity has not been clearly defined in IFC. As a result, the pattern must be configured as an assembly, with implicit pattern parameters. All these methods refer to the second approach of defining a hierarchical structure up-front by the exporting application and ensuring that the importing applications conform to it.

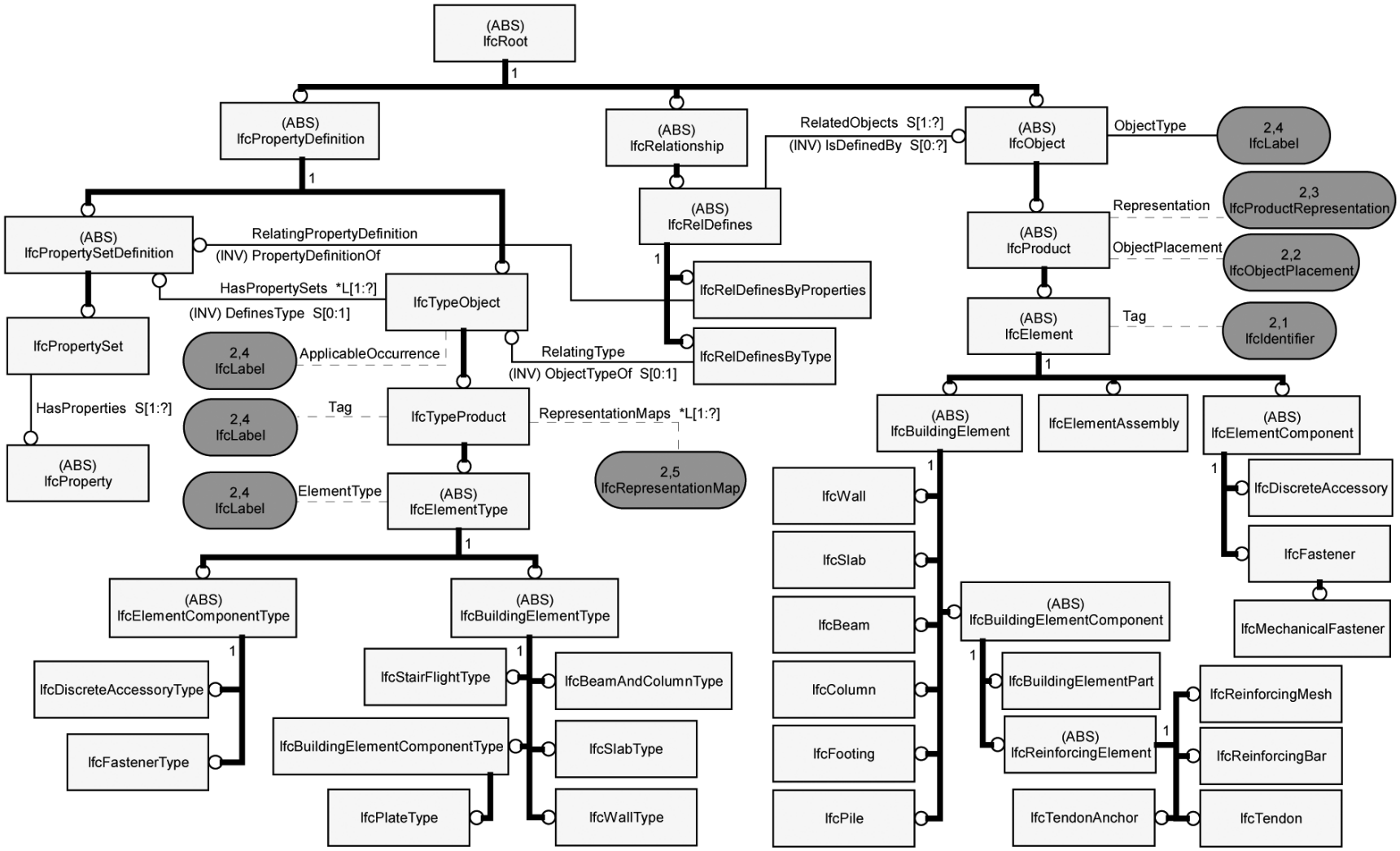

Figure 21: Hierarchy of object type concepts and relationships available in IFC [66].

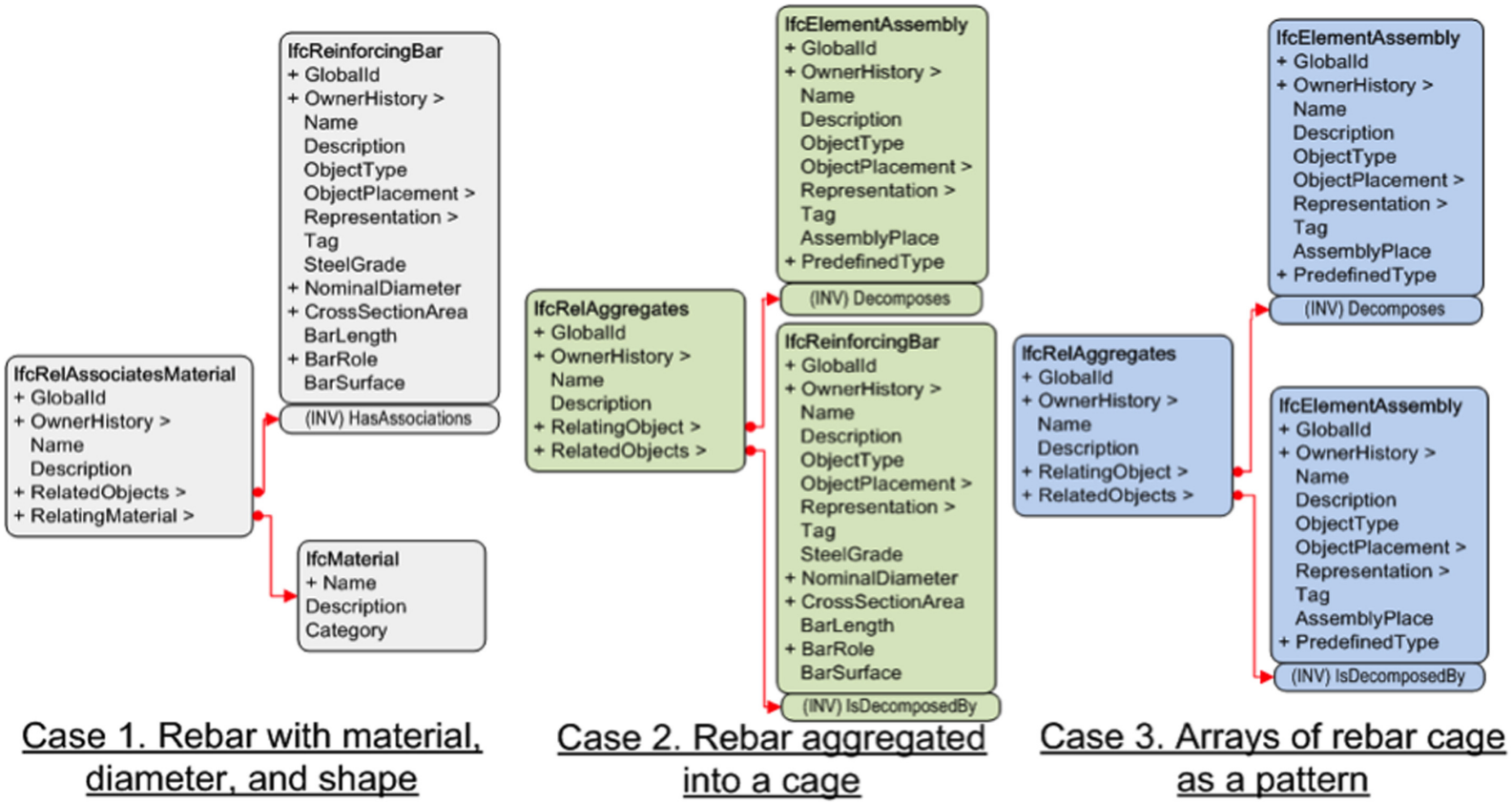

Figure 22: Different cases of reinforcing element aggregation, which can be used for type-instancing [119].

A third approach considers using a *multiple-inheritance* type approach. In this approach an instance obtains its defining features from multiple *type* instances, rather than from a 1:n style hierarchy. For example, a rebar instance may have a typical shape, a typical material, and a typical diameter, each of which may be inherited by other rebar as needed. One incentive for typing in this case is economy of file size. IFC does not currently support multiple inheritances, but a work-around is being commonly done in most IFC files by most export routines. Further, it can aid interpretation on the receiving end if the receiving software does 'understand' rebar shape types. In this way, if the instance-type relationship between rebar instances and *IfcShapes* is mandated, where shapes can have names, importing will be easier than if it relied on the importing software to scan and group the shapes. Figure 23 (a-d) [105] illustrates the reinforcement and tendon pattern definitions. An added benefit is that this approach is flexible, making it possible to represent multiple dimensions of typing in a single export model. The IFC has great flexibility to support multiple representations of the third type. This is due at least in part to the weak typing in the object-oriented programming sense (not typing in the sense we have used it above) that is characteristic of the IFC schema. Even though IFC is based on EXPRESS language, which supports multiple inheritances, IFC does currently not allow multiple inheritances in its schema. Making IFC fully compatible with ISO would be a long-term goal to achieving a better exchange mechanism. As a first step, the background meaning about the IFC objects and types that are currently implicit should be made more formal and explicit.

Type-Instance Summary: IFC is a *weak- (or loose) typing* system and provides multiple ways to type objects, thereby allowing great flexibility to support multiple representations. In order to avoid ambiguities in model exchanges it is imperative to make the SEMs a strongly typed system. Such a strongly typed SEM lattice on top of a weakly typed IFC schema can be the solution to achieving successful model exchanges.

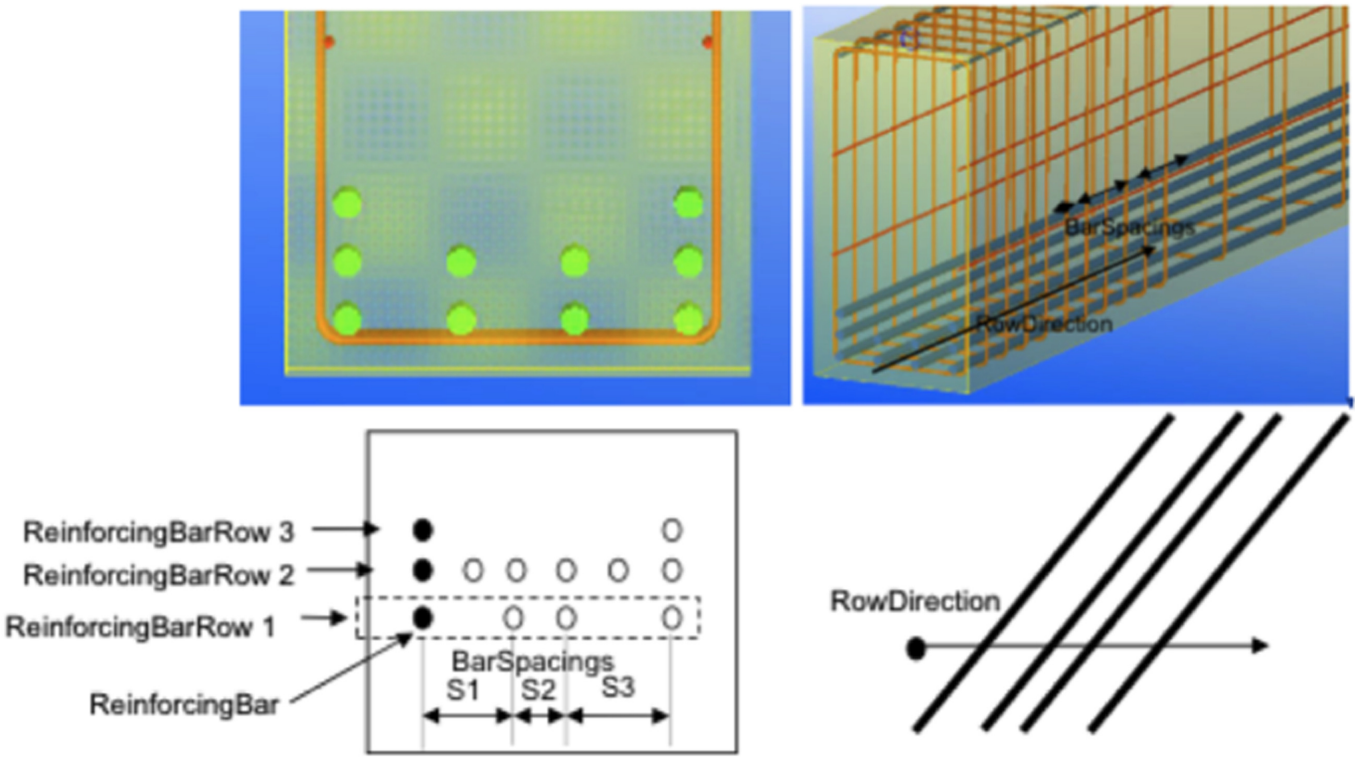

Figure 23: Reinforcement and tendon pattern definitions [119].

5.3 Classification Schemes

BIM tools provide another mechanism to structure their data - use of classification schemes. A classification scheme can be a standard and agreed manner to structure the domain information. Examples of *Construction Information Classification Systems* (CICS) are MasterFormat [88], UniFormat [28], Uniclass, etc. Classification schemes are implemented as a flexible and informal method at the software user level as compared to typing, which is more formal and is implemented at the programming language level. Classification schemes such as *piece designs, piece marks, standard cross sections, shared geometry, etc.,* are examples of this approach. Although classification schemes can be defined in different ways, these are not clearly defined in the IFC. For example, consider the classification scheme used for precast pieces. A *precast piece* is defined as a specific occurrence of a prefabricated concrete beam, column, spandrel or any other part of a building. It has a location that is defined within the building's coordinate system. These occurrences can be referred to as the instances and independently do not have their own geometry. However, in some cases minor local adaptations, such as a specific hole blocked-out for a pipe to pass through the piece, the geometry can be modified locally. A *piece mark* is a unique design for a precast piece. In fabrication practice, it is very common to have multiple pieces that are identical to one another. Instead of designing and drawing every individual piece, the pieces are represented as a group by a single design, called a piece mark. Thus a piece mark is a collection of pieces. Such an approach is used in steel, curtain walls, and most shop-fabricated objects. The *piece type* is the parent object that is commonly produced by a precast fabricator. For example, a fabricator may maintain a catalog of typical hollow-core plank sections with standard pre-stress tendon patterns. A *piece profile* is a cross section shape definition that can be commonly used for multiple piece types. They can be defined in libraries or catalogs and are used as the basis for multiple piece type occurrences where extruded solid geometry is used. A more detailed discussion on such parametric profiles can be found in the Geometry section (Section 5.4) of this chapter. A designer could choose the same piece type for multiple piece marks, and hence multiple piece marks would share a common piece type.

In an exchange, the properties of piece marks can be detailed in one of two ways: i) by

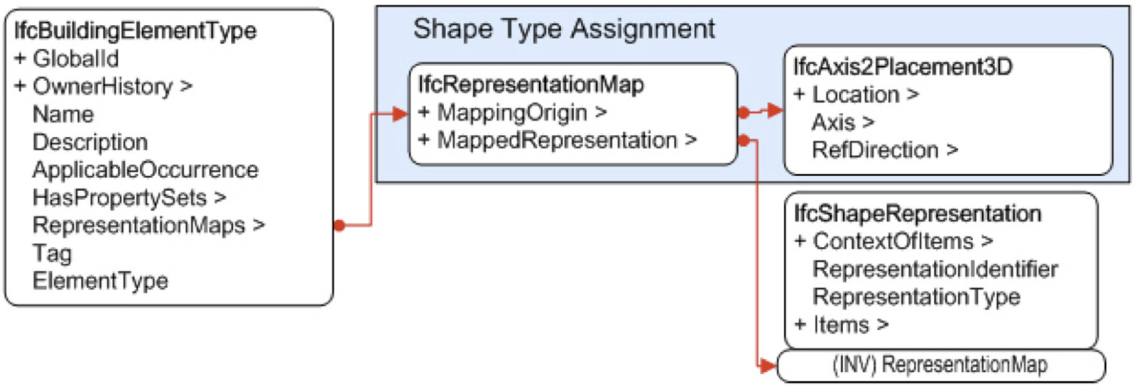

Figure 24: A building element with mapping to geometry using a shape representation object.

reference to an external data library that is shared among all parties to the exchange in advance; or ii) by carrying explicit property set values for the piece type and mapping its own geometry, rebar arrangement and pre-stress tendon placement pattern. It is assumed that the geometry is defined at the type level and only the location and the orientation information are defined at the instance level. This arrangement exploits the fact that the geometry of objects can be defined using a shape representation object, that is separated from the object itself. Moreover, multiple objects can refer to a single shape representation object. Figure 24 illustrates how a shape representation object is used to map the geometry to multiple objects. The classification scheme described above for a precast piece was proposed as one of the approaches, taking into consideration two major factors/advantages, a) reduced file size - the majority of information for repeated occurrences of the same part would need to be contained only once in the exchange file; and b) the increased utility of the exchange data in the importing application. The second reason is a consequence of the fact that the exchange file already groups identical or similar objects, which is important for most of the BIM functionality that involves editing or counting objects. In addition to precast pieces, classification schemes are recommended for reinforcing bars, lifting hooks, embeds, connection components, joints, finishes, materials and concrete mixes. Figure 25 shows a classification scheme developed for precast embeds on the basis of functionality and process level (plant applied, field applied, etc.).

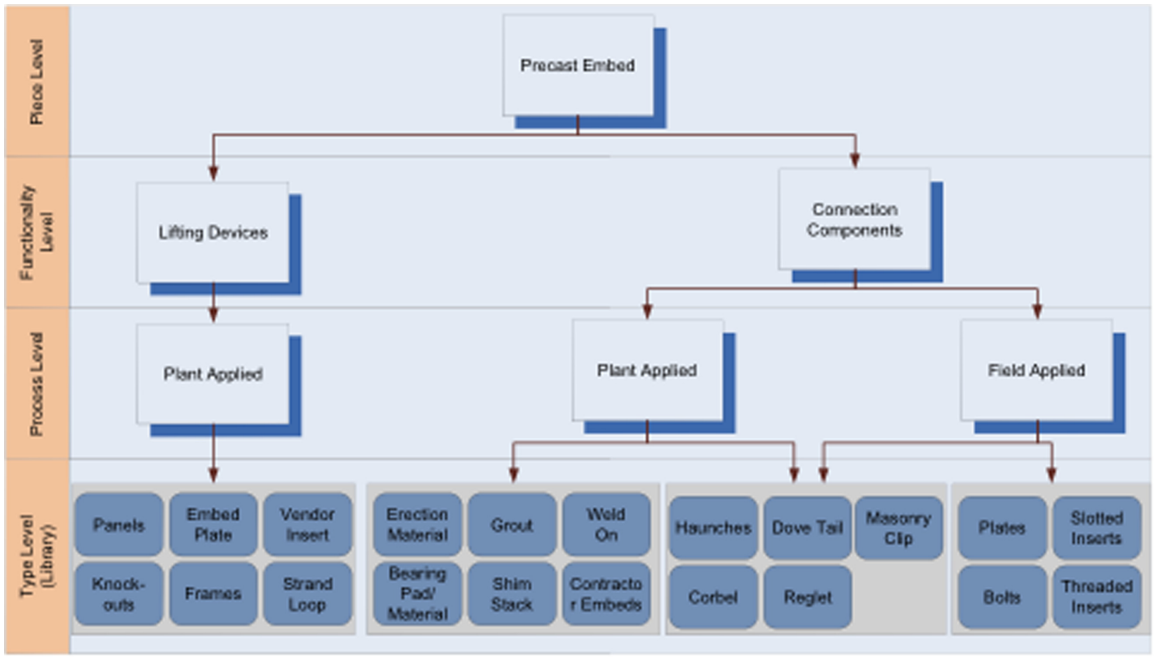

Figure 25: Classification scheme for Precast Embeds based on functionality, process and type.

Classification Schemes Summary: Classification schemes or simple groupings at user level provide an important means to structure the data in a model exchange. However, if either this classification is not included in the export or if the importing application does not support such classification of objects, then the intended semantics of classification is lost. Hence it is important to specify this classification in the model views and SEMs should support grouping of objects at different levels of meaning or functionality, thereby allowing model views to specify the classification schemes that are to be supported in the model exchanges.

5.4 Geometry

Exchanging geometry using IFC entities is possible in different solid modeling forms. Some of these forms include boundary representations (B-rep), extrusions and CSG. Figure 26 shows the solid modeling entities available in IFC. Consider as an example a manifold solid B-rep. This can be of two different types: The first type is to represent as a faceted B-rep in IFC release 2x3, or as an advanced B-rep in release 4. The construct for representing a

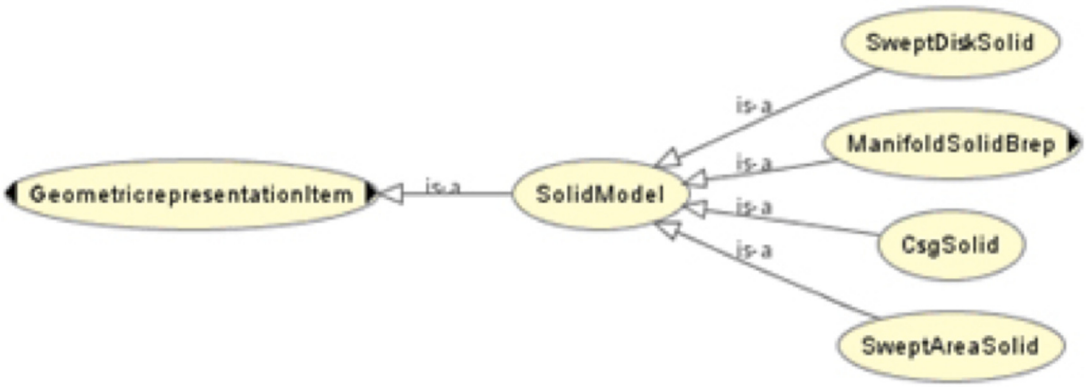

Figure 26: Solid modeling constructs in IFC.

face in advanced B-rep can be free-form geometry including NURBS, or B-splines. Another form of representation involves definition of entities by procedural sweeping action on a planar bounded surface. This is called the swept-area solid and in special cases, such as rebar, a circular disk can be swept along a curve, called a directrix. Usually the swept area is given either by profile definitions and position in space. The other option, namely CSG, is to perform Boolean operations on shapes to obtain more complex shapes. CSG combines geometric, solid models based on B-rep or Swept Area or Disk or Half-Space and CSG primitives, and structural information in the form of a Tree structure. All these constructs can be used in different combinations to represent a parametric shape. However, in the case of round trip exchanges or two one-way exchanges, the receiving application should be able to logically interpret the design intent and the original shape composition; otherwise the original information is lost. This leads to the research question of *when is the requirement of using more than just boundary definitions justified?* This question needs to be answered based on the exchange requirement which should be specified in the model views. The following paragraphs present some of the arguments supporting the above statement about geometry requirement. Modelers need to specify what representations should be contained in building and building modeling. Different aspects of the building that need to be modeled usually require different geometric representations. Three main divisions can consist of

1. building components such as walls, slabs, columns, etc.,

2. abstract geometrical forms used for conceptual models, and

3. building spaces, which are often derived spaces, and are defined by the components that bound them.

The boundary representation is the foundation representation used to store and possibly exchange information. Building components generally require all three types (B-Rep, CSG, extrusions) of geometric representation [38] and these representations are embedded in all BIM design tools. Some of these geometry representations are illustrated in Figure 27. For example, in the case of two-way exchanges or two one way exchanges, the recipient needs to select the entity instances to be incorporated into the new model. This instance is exchanged back to the sender, in order for the recipient to be able to browse and interactively select the entities to be downloaded to his or her application. However, if the geometry is simple B-Rep, the recipient will not be able to obtain any detailed object information such as opening dimensions within a parent piece, edge conditions, or parametric values, etc. In such scenarios, there is a need for geometry to be exchanged in a rich object oriented manner with all parametric details so that additional knowledge can be inferred from it. Therefore, the exchange of more complex geometric representations is important to many specific applications. Some of the semantic issues identified in exchanging geometry information are as follows [38]:

1. Shape method - B-Rep, CSG solid, extrusion, or other sweep.

2. Shapes needed as fabricated, or as deployed: deflections, cambering, warping.

3. All unique or is some of the geometry shared? - profiles, features, connections.

4. Surfaces - approximated, faceted, tolerances.

5. General accuracy of geometry.

6. Need for control geometry: grids, control lines or surfaces, control points, or local origins.

7. Reference coordinate system: project, assembly, longitude-latitude.

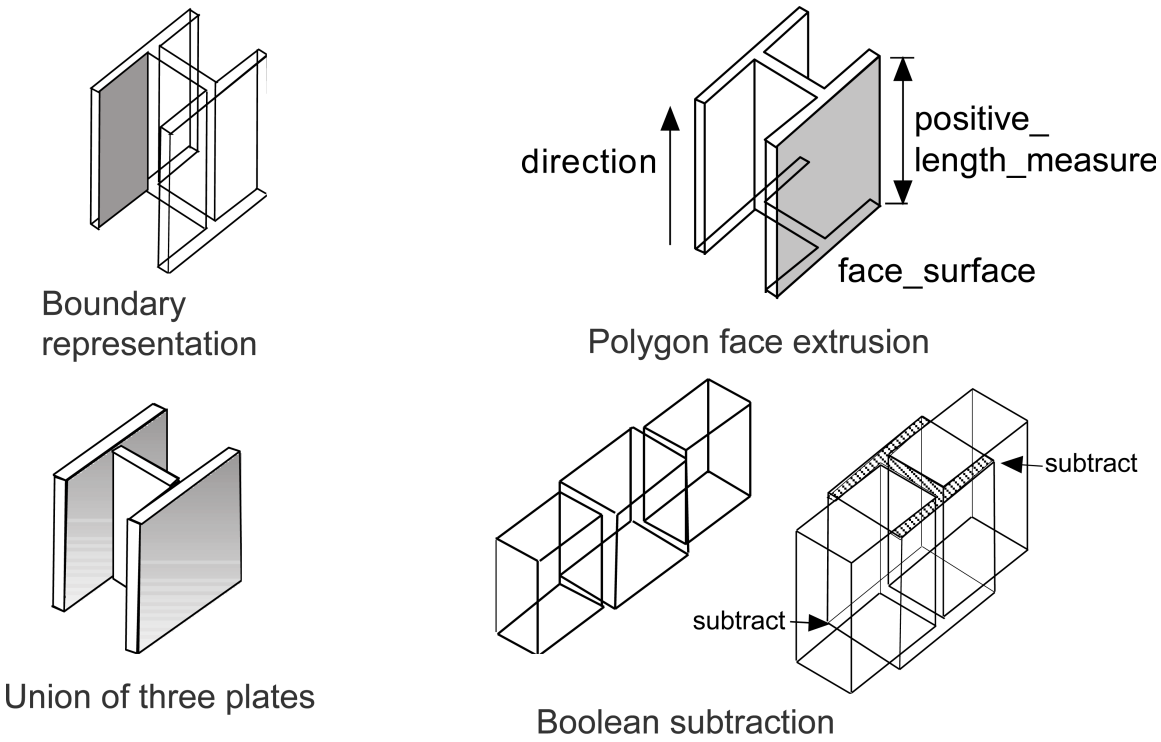

Figure 27: Shape methods available in IFC [38].

8. Performance model view: structural, energy, CFD.

In the case of building spaces, depending on the intended use of the space, various levels of accuracy regarding the shape of the space will be required. Most applications define a profile for floor wall intersections, and then extrude the profile vertically to a ceiling height. The area of the profile (floor area) is the most basic information of the space. Hence, we can say that different form-generating solid representations are used at various times during design. Each design potentially relies on a different geometric representation and different operations. These solid representations should be packaged in the form of SEMs with clear mappings to IFC schema. Such a SEM structure will help specify the exchange requirements clearly in the model views on the basis of SEMs. These *plug-and-play* SEMs for geometry allow building elements to be assigned to various geometry concepts based on the requirements without additional overhead. Further, the completeness and independence of these SEMs allows them to be reusable in various building elements. Figure 28 shows such a binding for a B-Rep SEM. Software developers need to implement this SEM only once,

and it could be assigned in any building element or object that requires B-Rep geometry, which means any future model views that use the same B-Rep SEM are bound to work.

Requirements for fabrication and detailing introduce more challenges. As an illustration consider a parametric representation of a precast Double Tee (DT). This representation requires the use of parametric profile definitions. Figure 29 (a and b) shows the *parameters* potentially needed to define the profile of a DT. In Figure 29 (a), the left and right stems show alternate configurations; either of these configurations can have chamfers or fillets at top or base. When the values of fillet parameters are greater than zero, chamfer values will be ignored. B-rep exchange is not sufficient if one needs to perform more than just clash detection or volume computation. Thus, when data is exchanged between applications, a range of additional semantics that help fully describe the objects in that exchange is required. Some of the benefits of parametric profile definitions include semantically meaningful and useful information and smaller IFC exchange file size. Different applications can rebuild these semantics at an appropriate level of internal detail, without necessarily replicating all of the finer details. Moreover, embedded components can be located relative to a parameterized profile definition. However, parameterized exchanges of this kind require both the exporting and importing applications to have explicit knowledge of the shapes exchanged. Some of these parametric shapes, such as the DT precast profile illustrated earlier, are specific to relatively narrow professional domains, and cannot be taken for granted.

Another example of such a parametric profile requirement is the case of precast hollow core (HC) slabs. Figure 30 shows the parameters identified for representing a HC slab. In all the cases, the cores are symmetrically distributed on either side of the plank center line, irrespective of whether the number of cores is odd or even. For planks with a center core with different geometry compared to the other cores, the spacing of center core is set to be greater than zero. When the number of cores is even, all Center Core parameters are ignored. Figure 31 (a-c) shows the detailed parameters for edge condition and the cores [105]. The *CoreTopRadius* and *CoreBaseRadius* parameters in Figure 31, can be derived and are therefore not required to be listed [105].

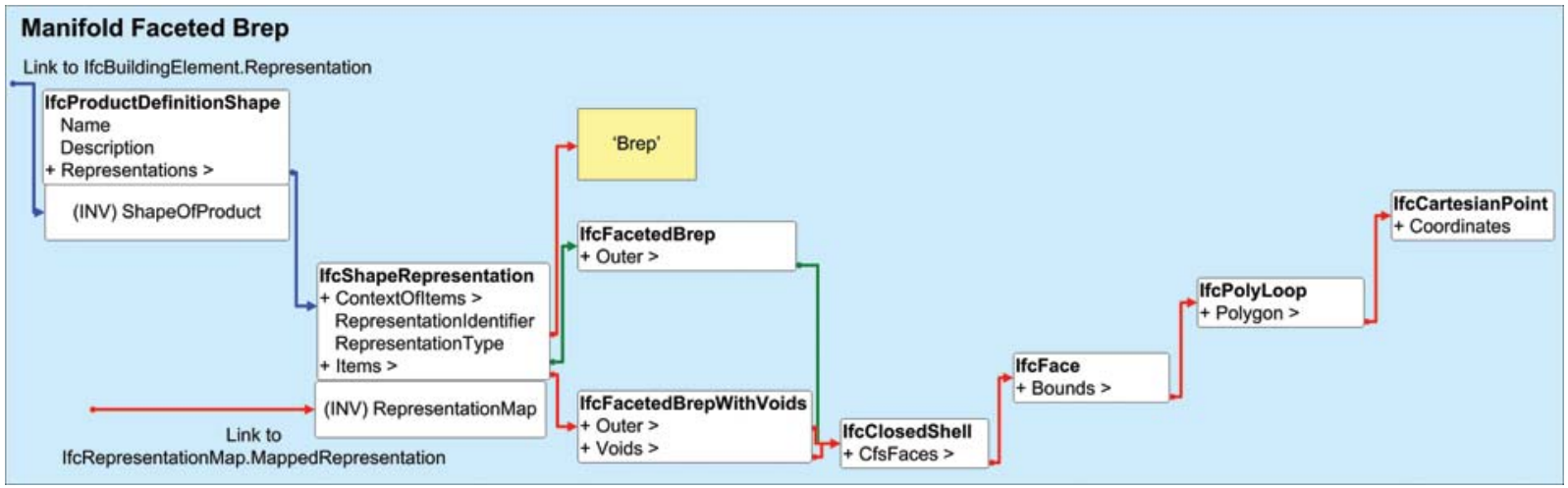

Figure 28: SEM structure for B-rep geometry.

Additional standard parameterized profiles are necessary for *AASHTO beam* types and other complex standard types. The IFC Technical Committee has made custom parametric profiles optional in Release 2x4, which enables passing *exotic* profiles without their specialized parameterization, while the specialty users can use these parameters to control geometry. However this leads to a disadvantage that a DT with an arbitrary profile can be edited into a form not allowed with the fully parameterized profile. In such cases, the roles of different users need to be clearly defined. For example, in the case of a precast model, the user is in an *Architect or Engineer* role and rights may be given to edit the geometry, but when a *Construction Manager* logs into the system he or she will be allowed to only edit information such as material types associated, schedules, cost or quantity information, etc., but not the original geometry. These are some of the semantics to be specified in model views.

Reinforcing bars were not exported in IFC files by most BIM applications until recently. IFC provides an *IfcReinforcingBar* entity and different semantic ways to export the rebar geometry information; namely as B-rep, or as extruded swept disk solid (known as *IfcSweptDiskSolid* in IFC2x3). By exporting geometry as a B-Rep, all required additional information is provided explicitly and the receiving application decides the import content and manner of import, etc. For example, a structural analysis application would ignore the geometry of individual rebar and express it in native objects in terms of the number of bars, spacing, diameter, etc. On the other hand, for clash detection, an importing application may decide to completely ignore rebar, if the volume of rebar is negligible. Refer to the discussion on relationships for more detail. Therefore we can say that translation is easily achievable, as long as receiving data need not be editable. However, this may lead to a bigger file size and no geometric operations will be possible. Full boundary representations are very expansive in terms of the file size as they require each surface and intersection to be defined explicitly.

Reinforcing bar can also be defined as a type with extruded geometry. This allows for multiple rebars to be instantiated from the same *IfcReinforcingBarType*. Multiple mapped

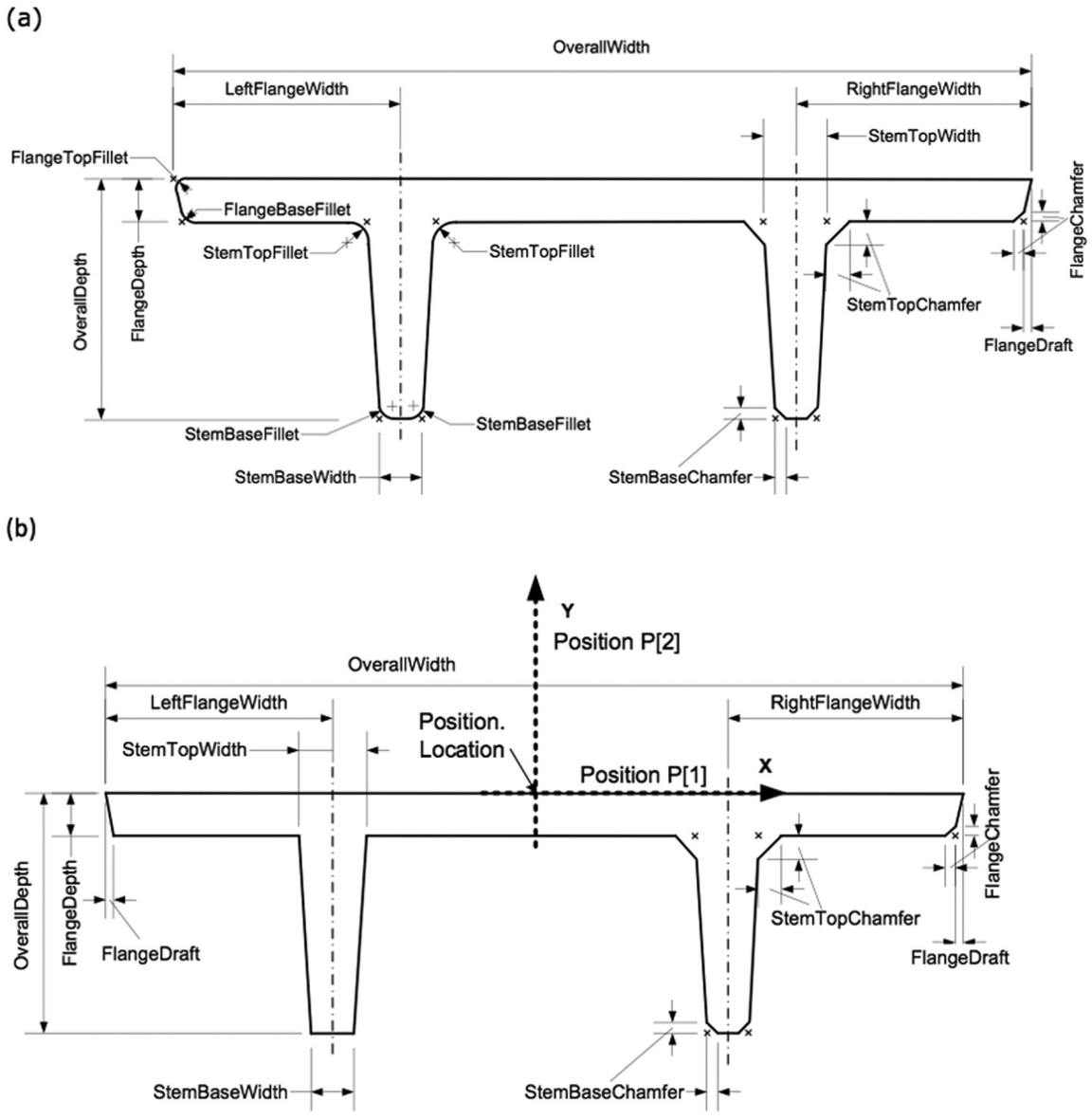

Figure 29: Precast concrete - Double Tee (DT) parametric profile definitions [119].

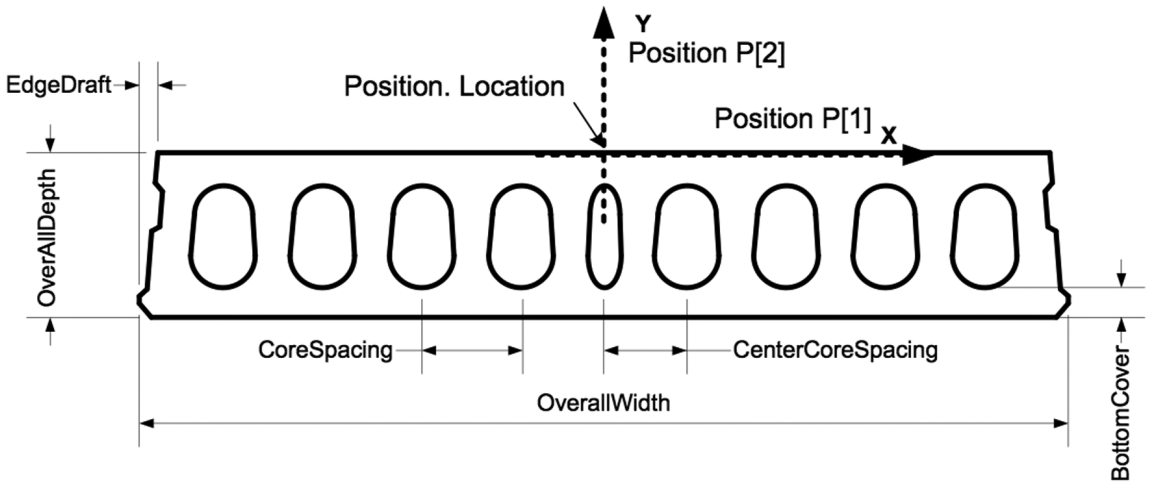

Figure 30: Precast concrete - hollow core parametric profile definition - Overall [105].

representations allow for several rebars to be represented by a single instance of *IfcReinforcingBar* and the number of mapped items corresponds with the rebar count in element quantity. However, this approach does not consider the case of rebar arrays, patterns or cages. (Refer discussion on array of objects in Section 5.4). Unless the representation scheme is specified and supported by the importing application, there is a chance that the associated semantics are lost, leading to misrepresentations.

Note: At the time of writing, IFC 2X4 (or Release 4) proposes a new entity called *IfcSweptDiskPolygonal*, which is a special case of *IfcSweptDisk*, with optional fillet radius. The assumption that export of rebar in extruded form is supported by most applications does not clarify many issues. For example, how the semantics of the type of solid modeling representation used can be embedded and also the use of entities like swept disk or swept disk polygonal, etc. The directrix can be represented using a *Polyline* or a *Composite Curve*. Since IFC 2X3 does not support fillet radius, rebar geometry was defined using swept disk profile and a composite curve as the directrix. (*IfcCompositeCurve* as the directrix with separate composite curve segments for the straight sections and the curved sections). Figure 32 illustrates rebar represented as extrusions and some of the limitations of an extrusion representation.

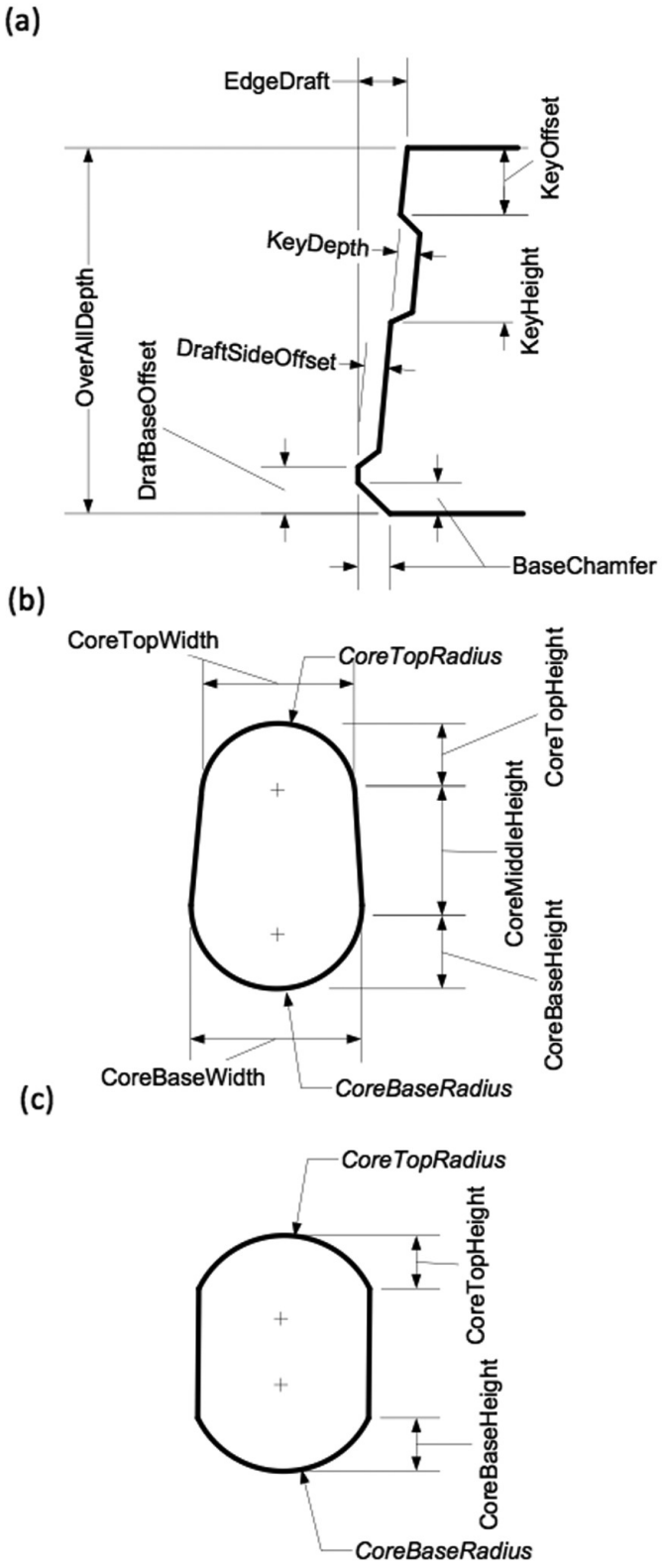

Figure 31: Precast concrete - hollow core parametric profile definition - Edge and Core definitions [105].

Geometric Representations Summary: Modelers need to specify what representations should be contained in building and building modeling. There is a need for geometry to be exchanged in a rich object oriented manner with all parametric details so that additional knowledge can be inferred from it, for many specific applications. These solid representations should be packaged in the form of SEMs with clear mappings to IFC schema. Such a SEM structure will help specify the exchange requirements clearly in the model views on the basis of SEMs. These *plug-and-play* SEMs for geometry allow building elements to be assigned to various geometry concepts based on the requirements without additional overhead. Further, the completeness and independence of these SEMs allows them to reusable in various building elements.

The different solid shape representations and their corresponding implementations using IFC present challenges, as discussed above in this section, that need to be addressed for meaningful model exchanges.

5.5 Relations and Rules

The IFC schema does not determine the behavior of entities within applications, apply parametric constraints or fix behavior, such as cleaning up wall corners, etc; this is at the discretion of the internal logic of each application. The condition of rebar and other embeds within concrete elements is similarly not dealt with in any manner that determines whether or not their volumes should be subtracted from the host element. The volume of concrete is the volume of the aggregate piece minus the volume of its embeds. Correspondingly, the weight of the concrete overlapping with the embeds must be subtracted to get the total object weight.

Two shapes can have one of three following relations:

1. Disjoint: the objects do not occupy the same space - anywhere. (A special case is where they share a surface, which could be treated separately.)

2. Nested: one shape is completely inside of the other - everywhere. (The special case applies here as well)

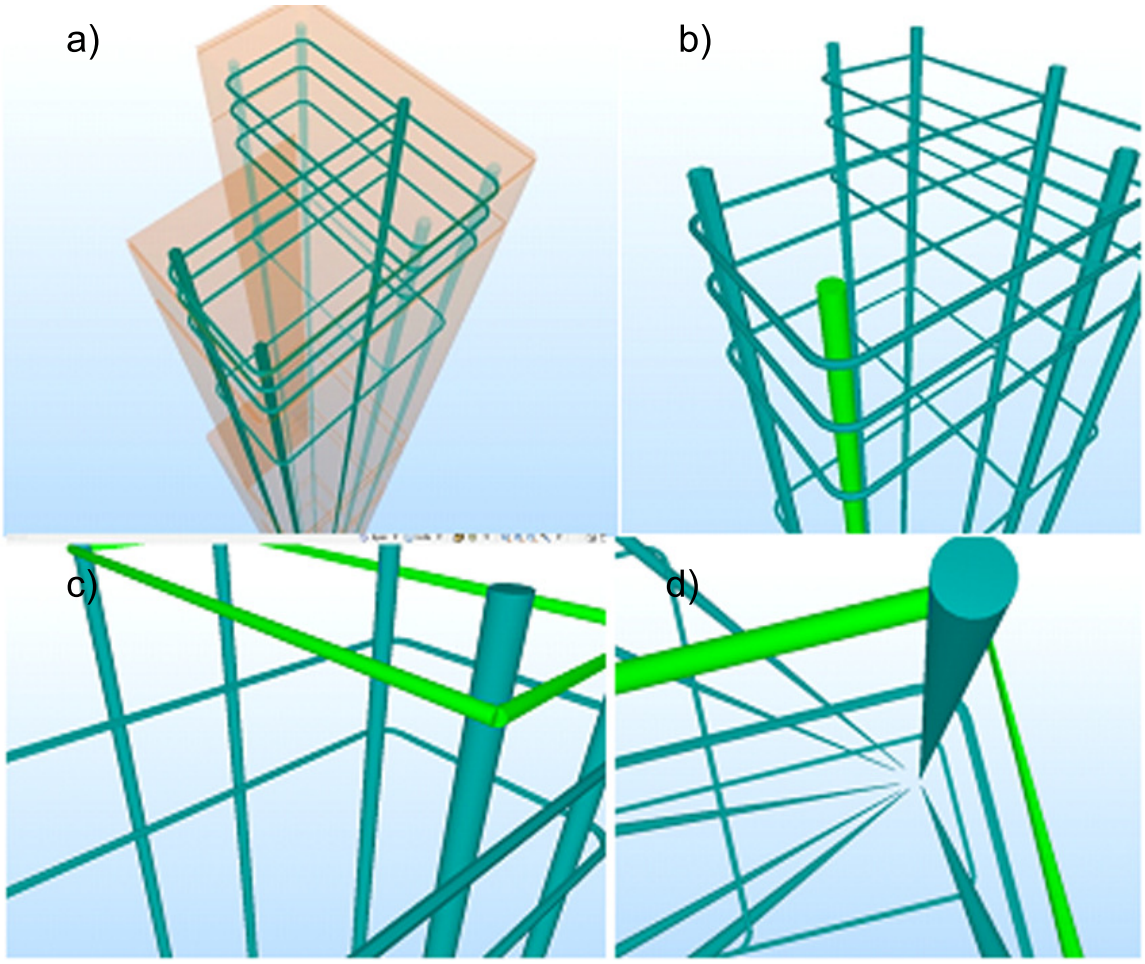

Figure 32: Representing reinforcing bar with a) B-Rep geometry with non-circular cross-section, b) extruded geometry, c) errors -corners are not rounded (orthogonal joints if we use *IfcPolyline* as directrix, d) errors - the end of line segments getting tapered.

3. Overlapping: one shape is partially inside and partially outside the other.

For example, consider the relationship between an *IfcBeam*, say, and an *IfcReinforcingBar*, implemented using *IfcRelAggregates*. The simplest approach accepts that this implementation is satisfactory. It does not question whether the volume of the rebar should be subtracted from the volume of the concrete to the applications that will manipulate the data. As long as the basic condition of semantic identification of the parts is met, (i.e. the rebar is identified and recognized as a rebar) it is part of (defined by *IfcRelAggregates*) of the beam. Each application can then decide whether or not the volume should be subtracted according to the task at hand and the degree of accuracy required, for both soft and hard clash checking. This semantic condition has been addressed in release 2x4 of IFC with the addition of *IfcRelInterferesElements*. This has been defined as a one-to-one relation between two objects, which again leads to the need for addressing aggregations of rebar, not single pieces. To summarize, there are multiple solutions to this problem including the use of *IfcElementAssembly* for the rebar and defining the relationship, or relating every piece of rebar to the beam it is in. Moreover, in certain cases it is necessary for a particular assembly to have an identity, that separates the assembly from the aggregation of its components. This is justified in the case of cast-in-place concrete, where connected volumes have resulted from overlapping concrete volumes [6]. This can also be considered when cases such as precast modules like prison cells, sanitary blocks, staircase sections etc. are to be considered. The geometry of these assemblies cannot be expressed in a simple manner as an aggregation of other object types. However, introducing new entity types, which was done with IFC in the past, will result in an explosion of element types. Hence, without clearly defined model views, it is impossible to infer knowledge from a model exchange based on context.

Behavior or function of the entity is another aspect to be considered when using relationships. A particular example is that of *feature based modeling*. This is a generalization of all feature-based modeling entities within IFC, which are existence dependent elements. This applies only to those elements used to modify the shape and appearance of the associated master element. It has to be noted that there is a distinction between using *aggregation* relationship, where components are treated as equal parts, and *features*, which

are subordinate parts of a parent element. For example, corbels are feature additions and provide a *Boolean addition* to a building element, such as a corbel or any other projection from the normal bounds of the piece. Use of relationship *IfcRelProjectsElement* to connect the feature (subtype of *IfcFeatureElementAddition*) to the parent element is required in this case. A recess or opening are subtractions and creates a void or opening element (subtype of *IfcFeatureElementSubtraction*) in a building element through the relationship *IfcRelVoidsElement*. Figure 33 illustrates the seam connection between two double tees. This connection is realized by means of a feature element subtraction (void). The connecting elements are also attached to the parent elements through an *IfcRelFillsElement* relationship. These are treated as semantic objects but their sole purpose is to modify the shape of the parent object, hence based on this behavior it should be invalid to use a simple aggregation relationship. Correctly implementing this rule and conforming to it by both the users/applications is the way to query or track the modifier at a later stage for editing purposes. Similarly, the placeholders for attributes or properties for objects are not strictly defined within IFC. For example, if a receiving application needs to discriminate between precast pieces and cast-in-place, rules need to be written to check placeholders such as name, description, object type or tag, since no strict guidelines exist for embedding these semantics in a formal manner. These should be defined in a model view and are helped by globally agreed upon conventions, for use across all MVDs. One of the options is to classify the piece into a precast or cast-in-place grouping. However, a more semantic approach is to aggregate them into a precast system or cast-in-place system. Such an approach will enable the return of actual quantities for queries by including the non-precast pieces (for example, steel embeds in a precast piece) as well, which are still a part of the precast system.

Another example of a business rule is the case of spatial containment of entities spanning multiple storeys in a building such as curtain walls, elevator shafts, multi-storey columns, spaces spanning multiple storeys, etc. In order to correctly represent and relate the multi-storey entity, the rules are as follows:

1. The multi-storey entity should be attached to one of the floors in the building and referenced by all the other floors through which it passes.

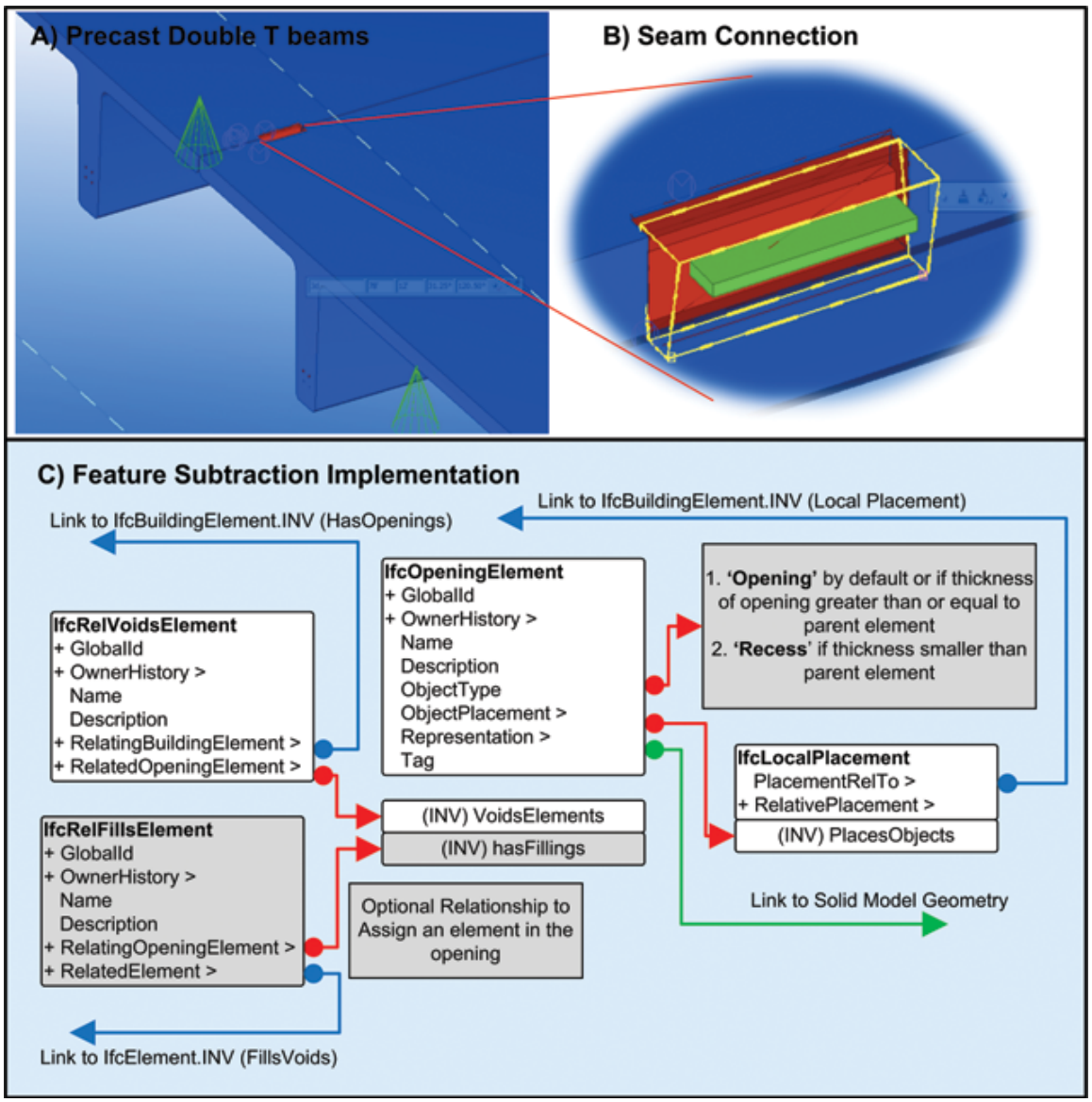

Figure 33: Feature based modeling. a) Double tees with seam connection, b) illustration of the seam connection with void feature, and c) ifc mapping for the void feature and the seam connection.

2. *IfcRelContainedInSpatialStructure* relationship should be used for attaching the entity to the spatial container.

3. *IfcRelReferencedInSpatialStructure* relationship should be used for referencing the entity in multiple floors.

4. Containment should a mandatory one-to-one relationship, meaning one building element can be and should be contained by only one spatial container.

5. Referencing is an optional one-to-many relationship, meaning one building element can be referenced in zero or many spatial containers.

Strictly conforming to such rules as provided in the model view will help the downstream application to not just display the geometry but also detailed object information. Moreover, when building models are modified, a mechanism is required to relate and track the piece information. IFC provides this functionality through the Global Unique Identifier (GUID) [80], which is a unique identifier throughout the software world. However, when data is transferred back and forth by applications that have differing native conventions, preserving the GUIDs becomes an ordeal and a mechanism for performing this GUID preservation has not yet been widely implemented. Such tracking is important, for example, in the case of a precast concrete pieces where design is sent from the engineer to the detailer (structural coordination model). The fabricator performs the job of piece detailing and sends the model (fabrication model) back to the engineer for structural review. A beam, its connections, and rebars in the beam, etc. all have GUID, but for practical purposes it is easier to merge these pieces into a single entity. This is another example where conventions are needed for tracking pieces when they are aggregated. At a different level, the use of GUID and Owner History, as defined in IFC, creates an overhead to the software models. For example, GUID is a property of *IfcRoot* and all building elements inherit this property, including entities such as *IfcReinforcingElement*. This forces each instance of rebar to have a GUID rather than assigning a group of reinforcing as an element and then assigning a GUID. This issue arises in the case of structural fasteners, holes, etc. where individual parts do not require a GUID. Allowing repetitions and arrays of objects breaks the established

notion of GUID within IFC. Implementing a multiple inheritance structure for components such as *IfcReinforcingElement* can potentially solve the overkill of GUID requirement. The idea of building SEMs on top of the IFC schema can help to remove some of the ambiguity associated with GUID related issues. SEMs can be defined in such a way that GUID can be mandated to be one of the attributes if the parts cannot be aggregated together into repetitions. In other words, if there is need for a GUID, then that part cannot be aggregated into repetitions. Both of these conditions are necessary to validate the inheritance of GUIDs.

Relationships and Rules Summary: IFC doesn't specify the semantics associated with relationship classes. Issues such as array of objects, GUID tracking, overlapping volumes, etc., present challenges in model exchanges and the proper interpretation of what is exchanged. Multiple inheritance can be a potential solution, however the upward compatibility of data schema needs to be analyzed.

5.6 Results and Recommendations

There are plans to elevate IFC into an ISO compatible standard (ISO/IS 16739) in the future [69, 22]. However, until then, it would remain an industry-led undertaking to provide model exchange capabilities to AEC-FM industries. IFC is a rich model that addresses the needs of different applications and provides a variety of ways to define the same part of a building. Hence additional layers of specificity such as model views are required for IFC implementations. This brings to the forefront the need for a more logical framework to specify model views. The number of research and industry-based initiatives to develop model views in different areas underlines this need. The PCI team utilized the IFC Solutions Factory, which is a web-based repository of bindings and model view development efforts that are being pursued in different parts of the world. Table 5 tabulates some of this research and their respective target model exchanges [13]. A number of these areas have overlapping information; however, lack of strict definitions makes it impossible to reuse many existing bindings, which adds to the overhead for software developers. For example, precast and cast-in-place concrete should have different sets of model view definitions as they involve different set of processes for erection or casting of the piece, but the reinforcement requirement could

be largely the same, and should share common bindings. This implies that whenever in-place concrete model views are developed, there is a potential for reuse from the already defined precast model views. The introduction of SEMs is seen as a positive development in terms of their intended re-usability and modularity. Figure 10 shows a precast connection component as an example for SEM definition. However, the idea of a SEMs and the granularity of the SEMs need to be clearly defined. Hence, the proposal for a formal and rigorous framework to define the SEMs is justified. Moreover, IFC is an extensible data schema, where new extensions to the schema are proposed and accepted based on new business requirements. It is typical for a gap-analysis to be performed and new extensions to be proposed during the development of model views [105]. There is criticism that some of the extensions are done in an ad-hoc manner [75]. This claim is in fact justified by the number of IFC entities that are introduced and then deprecated, while moving from one version of IFC to another.

The issue of semantic robustness of model exchanges using IFC, illustrated by the varied examples in this chapter, needs to be seriously considered for advancing interoperability within the AEC industry. The discussions provide insights into the conundrum of embedding semantic meaning in exchange data, with specific emphasis on the type-instance structure, classification, geometry, relationships, and rules. Based on the work conducted in developing the Precast National BIM Standard and further analysis of the past and present work in this area, a set of recommendations are presented here. These are grouped into categories.

5.6.1 Model View Definition

- The MVD development process needs to be transitioned from the current manner to a more rigorous and consistent framework and/or methodology. Some steps for improving the quality of information in the IDM phases of MVD development are outlined in [36], and *A Guide for Development and Preparation of a National BIM Exchange Standard* [33].

- The semantic meaning of IFC entities, relationships, attributes, and property sets, needs to be defined in a rigorous and formal manner with strict guidelines. Implementation of SEMs based on formal semantic guidelines can help in achieving a

Table 5: A list of current model view development initiatives in progress using IFC [13].

Exchange Model	**Organization**
Architectural design to circulation/security/analysis	US General Services Administration
Architectural design to landscape design	CRC for construction innovation
Architectural design to quantity take-off - level 1,2,3	Virtual Building Laboratory; German Speaking Ch.
Architectural design to spatial program validation	US General Services Administration
Architectural design to structural design and to structural analysis	(VBL)Virtual Building Laboratory @ TUT
Architectural design to thermal insulation	(VBL)Virtual Building Laboratory @ TUT
Architectural programming to architectural design	BuildingSmart International
Basic handover to facility management	German Speaking Chapter
Concept design BIM 2010	US General Services Administration
Design to code compliance checking	International Code Council
Design to energy perf. analysis	Building Smart Alliance, North America
Design to quantity take-off	Building Smart Alliance, North America
Extended coordination view	IAI Implementers Supp. Group
Extensibility	(VBL) Virtual Building Laboratory @ TUT
Indoor climate simulation to HVAC design	Helsinki University of Technology, HVAC Lab
Landscape design to road design	CRC for construction innovation
Precast Concrete Exchanges	Precast Concrete Institute
Road design to landscape design	CRC for construction innovation
Space requirements and targets to thermal insulation	Helsinki University of Technology, HVAC Lab
Structural design to structural detailing	Applied Technology Council

uniform mapping to and from the internal objects of BIM tools and IFC entities and relationships.

- Standard criteria for defining the SEMs proposed here should be documented to avoid various research and development teams generating varying implementations. Such a standard approach will help in reuse of SEMs thereby resulting in the reuse of MVDs itself.

5.6.2 Semantic Exchange Modules

- SEMs, once tested and implemented, can provide a mechanism to generate model views directly from exchange requirements. This is a novel idea and has yet to be explored.

- There appears to be a huge potential to reduce the current model view generation - implementation cycle time of 2-3 years to a more practical 4-6 months initially and to an optimum of few hours by following a modularized approach using SEMs.

5.6.3 IFC Ambiguities

- There should be flexibility in defining the type-instance structure based on the context and nature of an application. A multiple-inheritance structure can be the long-term solution for achieving this flexibility. However the study of the upward compatibility of the schema needs to be propelled by further research. This is an important research issue, to be addressed when IFC is made fully ISO compatible.

- IFC is a weak (or loosely) typed system and provides multiple ways to type objects. In order to avoid ambiguities in model exchanges it is imperative that the SEMs are modeled as a strongly typed system. Such a strongly typed SEM lattice on top of a weakly typed IFC schema can be the solution to truly realizing successful model exchanges.

- Classification schemes can be used to group entities and structure the data in a model exchange thereby reducing the file size of model exchanges. This also increases the

utility of the exchanged data in the importing application due to the fact that exchange already groups identical or similar objects. This is important for most BIM functionality that involves editing or counting objects and such semantics should be specified in the model views.

- Editable geometry is still not achieved in model exchanges; however, the use of parametric profiles, as shown in Figures 29, 30, 31, can provide this feature to a certain extent.

- The level of detail requirement of the model views and the model progression is another important topic to be taken up by the industry

CHAPTER VI

ONTOLOGY DEVELOPMENT

This chapter explains what the requirements of a Precast System Ontology are and how they can be specified. Knowledge is modularized in small, manageable pieces that can be reused. These building blocks are called the Engineering Ontologies, and are formed from super theories of mereology, topology, and systems theory. An application ontology called Precast System Ontology is developed and describes the requirements for building systems comprised of precast pieces. Precast System Ontology forms the basis for defining the SEM library for exchange modules of precast systems

6.1 Assumptions and Requirements

An Ontology is a formal specification of a shared conceptualization. Formal definition is essential for ontology reuse. Clarity, coherence, extensibility, minimal encoding bias, minimal ontological commitment [51] are features of a well designed ontology. The Ontologies described in this thesis have been specified in the Web Ontology Language (OWL) using the Protege ontology editor. Some of the requirements and assumptions for the ontology are presented below.

6.1.1 Formalization Requirements

Alignment to an ontology will help to exclude the terminological and conceptual ambiguities due to unintended interpretations. Hence, the first and foremost requirement of this research is to raise the IFC semantics to a formal level, as illustrated in Figure 14. A formal language needs to be selected for representing IFC Semantics. A formal language that represents IFC semantics should, as a minimum:

- be compatible with the EXPRESS schema of representations
- be formal as well as easy to understand and

- provide mechanisms for automated reasoning

6.1.2 Normalization Requirements

A second and important requirement, which was identified during the current model view work, is the need to avoid redundancy and rework in terms of development and testing [117]. Hence, SEMs should be generated following strict guidelines so that they are testable and standalone. For new MVD development, these should be in a plug-and-play form. Retesting, which is expensive and time consuming, should be avoided. Some of the requirements for normalizing are:

- Complete and standalone SEMs
- No broken links or references
- Retrievable queries
- Taxonomical structure or hierarchy
- Lattice of primitive or composite terms together with associated definitions
- Explicit definition of semantic relationships among terms

6.1.3 Application Requirements

The ontological system and SEM structure developed should support querying of content-based data from a product model or from a model server. The current terminological and conceptual ambiguities need to be removed by the formal structure so that it minimizes semantic mismatch during querying for various applications. For example, queries should be able to provide data on whether it is geometry based or numerical analysis based etc. A future application would involve supporting a transaction based checkout and check-in of partial product models. This involves inter-operability among humans and machines and is a perfect application for ontologies.

6.1.4 Corpus Constitution

In terms of the modeling criteria, a corpus or body of knowledge needs to be constituted. Entities are selected from the available domain-specific documentation according to the ontology requirements. The Precast NBIMS Model View is selected as the corpus in this case. Once the ontology is developed and validated, it can then be applied to other domains as well. Appendix B lists all the requirements for the precast system in terms of information groups, attribute sets, and attributes. This corpus is used as the basis for developing the Precast System Ontology. It should be noted that this thesis aims at capturing knowledge about the physical model of buildings. Functionalities such as structural analysis that are part of the analytical model are left out of this research.

6.2 Ontological Definitions

The IFC data model is envisioned for different services across various domains, whether for a specific data exchange between a precast detailer and structural engineer, or between an architect and MEP contractor. The descriptions of services should be formulated according to an ontology in order to support the automation of service related task. These types of services can be defined as some of the uses of the IFC data model. Main criteria for development of an ontological structure is knowledge sharing and reuse, hence it is evident that the ontology developed will be large and complex. A *divide* and *conquer* approach [19] is followed in developing this ontological structure.

IFC entities and relationships are expressed in the form of an ontological structure [120]. OWL2.0, which is the standard for semantic web, is adopted for this representation. An extensive ontological structure that contains explicit semantic definition of entities (classes), relationships (properties), attributes and types are generated. This structure is saved in a database (model repository/server) that supports IFC and semantic rules.

So the key features of the ontology system can be defined as follows:

- OWL syntax based
- Domain knowledge captured and extensive semantic definition of entities provided

- Can be visualized as tree structure or graphs and read as XML or like a normal relational database supporting IFC
- Users can extend the ontological structure for future revisions

6.2.1 Define Ontological System

IFC in itself provides a very good hierarchy of concepts which are structured in a manner ideal for the application of ontological definition. Moreover, it is rich and redundant. This implies that there is more than one way of representing the information to be exchanged. A formal and strict definition in the form of a set of axioms and constraints based on predicate logic can be considered as a tool to facilitate machine understanding. Reasoning algorithms that run on such logic can be developed and utilized by importing and exporting applications to derive and interpret the information in an intelligent manner. Accomplishing this task requires the concept definitions and constraints (business rules) to be represented in an ontology language.

Protege and Web ontology languages (OWL and OWL2) are the tools that are used to represent knowledge in a structured and reasonable way. OWL has been developed by the W3C consortium as the enabling language for semantic web [64]. Protege is a robust development environment for knowledge-based systems [47]. The key feature of these tools is that they are based on Description Logic (DL), which is formal with well- defined semantics. A DL model consists of a domain (IFC entities) and an interpretation function (mapping of relationships to IFC entities). Importance is given to the relationships (subsumption, equivalence etc) between the entities. DLs help in sharing information defined by ontologies without any misinterpretation of their terminology and meaning.

Ontologies can be classified on a spectrum of varying degrees of semantic precision with simple glossaries of terms on one end of the spectrum and rich ontological theories on the other. The hierarchy of ontologies used this research is shown in Figure 34. This forms the *Conceptual Model*. As the degree of semantic precision increases, so does the complexity of computational reasoning. An effective method to maintain computability is to separate representation and reasoning.

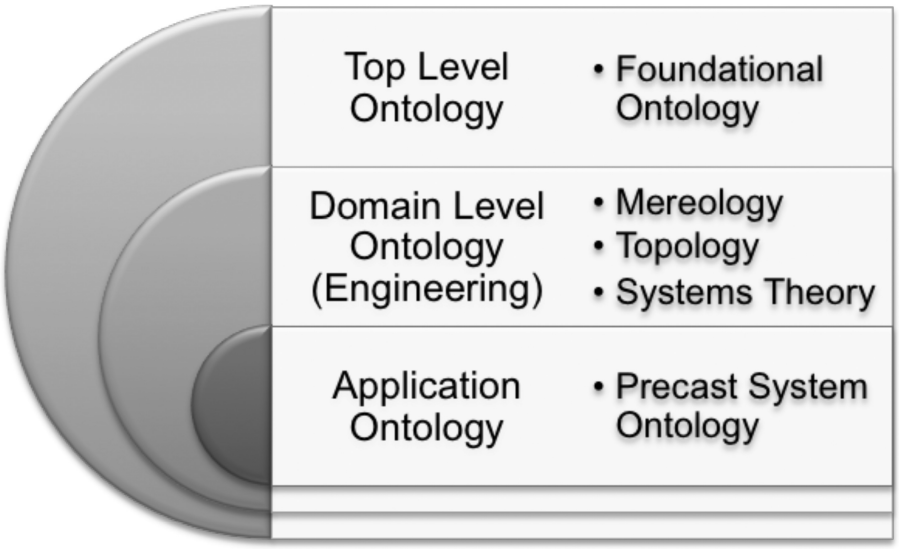

Figure 34: Conceptual Model: Ontology Hierarchy for Precast System Ontology.

A sound base is important for building any hierarchy. This is achieved in this research by structuring the ontologies on a *foundational ontology* such as DOLCE. This is the most abstract layer, introducing the basic modeling concepts and generic design guidelines for the construction of actual ontologies. The second layer consists of *super theories* such as mereology, topology, and systems theory, that is reusable modules according to which ontology is organized. The final comprises of application specific ontologies such as structure of object (precast specific material, geometry, etc.) and properties. The application layer refines the ontology to be used for Precast System by adding classes and relations for practical application of ontology.

Foundational Ontology: Foundational Ontologies are specialized logical theories not limited to particular domains and developed with the intention of characterizing explicitly a viewpoint on the reality [16]

A Foundational Ontology for AEC: Foundational ontologies (FO) is aimed at capturing formally the intended meaning of the adopted language. Integrated Project Delivery in AEC requires applications from domains as varied as Architectural, Structural, MEP, Energy or Spatial analysis etc to interoperate. Hence, domain knowledge is divided into small, manageable pieces with strong relationships to the above application domains. These

are specified in separate ontologies that form the building blocks of the larger ontology. These are known as **Domain Ontologies** (in this case **Engineering Ontologies**). To preserve the semantic meaning across these domains during model exchange and to facilitate interoperability there is a need for a foundational ontology. This should be adapted for AEC. Some of the foundational ontologies in the literature are the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), General Formal Ontology (GFO), the Object-Centered High-level Reference Ontology (OCHRE), and the Suggested Upper Merged Ontology (SUMO) etc [87]. A foundational ontology heavily based on DOLCE and adapted for AEC is recommended in this research. **Application Ontologies** are specific to a problem domain or application. In this case, an application ontology for precast pieces is developed. The application ontology evolves from a transformation of the fundamental and domain ontologies into a representation that is consistent with the syntax and semantic constraints of the programming language (in this case IFC). This can be directly mapped to the software implementations.

Top Level Ontology is a abstract notion covering high-level philosophies and is not very useful for an implementation. Domain level ontologies are specific to a particular domain, in this case Engineering. Based on the foundational ontology, the topmost layer is classified into different categories as shown in Figure 35. Thing is the abstract root of all the categories. The level-1 categories are Objects, Occurrents, Quality, and Relations. Table 6 lists the sub-categories of level-one entities. Objects are those, which are present wholly at any point of time. These can include Physical and Non-physical entities based on whether they have direct spatial qualities. Physical entities can be again divided as Amount of matter, Features, and Physical objects. Physical objects can be again Object instance or Object type. Non- physical objects can be either Mental and Social. Figure 35 illustrates the hierarchy of the Object structure. An entity can be categorized as an Occurrent if it exists at more than one moment and its temporal parts can be determined only relatively to time. These are again subdivided into Event, State, and Process.

Qualities are inherent to entities and can be seen as the basic properties that we can perceive or measure: shapes, colors, sizes, sounds, smells, as well as weights, lengths, etc. Qualities

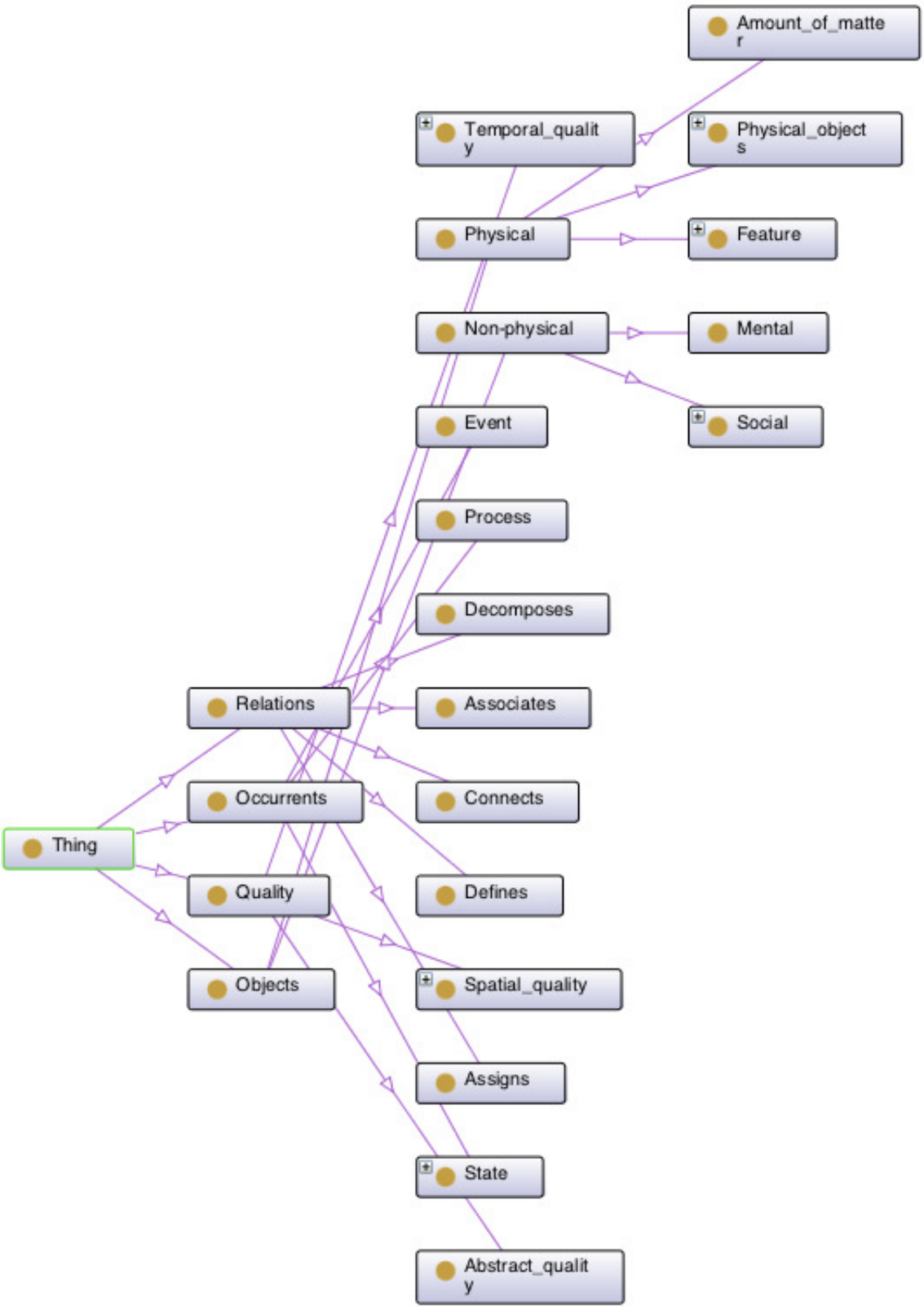

Figure 35: A Foundational Ontology for AEC-FM based on DOLCE.

Table 6: Sub categories of level-one classification of entities.

Level-one Categories	Objects	Occurrents	Quality	Relations
Level-two Categories	Physical Non-Physical	Event State Process	Temporal Physical Abstract	Assigns Associates Connects Decomposes Defines

Table 7: Examples for the concept categories.

Concept Category	Example
Amount of matter	Some concrete, some water
Feature	A hole, opening etc
Physical Object Type Instance	 Beamtype Beam
Mental Object	A percept, sense of datum etc
Social agent	Contractor (person or organization
Event	Holiday
State	Approved
Process	Concrete pour
Temporal location	Location on site
Material	Steel, Concrete etc

can be temporal, spatial or abstract depending on whether they characterize an Object, Occurrent etc. Relationships can be divided into Associates, Assigns, Connects, Decomposes, Defines etc. based on their function. Table 7 gives examples for some of the categories of classification. Engineering ontologies can be seen as a *Lego model* for building application ontology [52]. In this case, we have components, connections, and system ontology, as the engineering ontologies. These are called the super theories and are explained in the following sections. Notations and symbols used to define the relationships are shown in figure 36.

6.2.2 Component Ontology: Formal theory of parts

The Component Ontology defined in this thesis is influenced by the theory of Mereology explained by Simons [110] and in PHYSYS [19].

symbol	meaning	symbol	meaning
$\supset$	logical implication	$\circ$	overlap
$\equiv$	logical equivalence	$\mid$	disjointness
$\wedge$	logical and	$+$	binary sum
$\vee$	logical or	$-$	mereological difference
$\sim$	logical negation	$\cdot$	binary product
$\exists$	existential quantifier	ι	definite description
$\forall$	universal quantifier	σ	general sum
$\ulcorner \ldots \urcorner$	scope of quantifiers	π	general product
$=$	equality	$\asymp$	connectedness
$\approx$	identity	$][$	disconnectedness
$<$	part-of	$\succ\!\!<$	external connectedness
$\ll$	proper part-of		

Figure 36: Logical notations for Engineering Ontologies [19].

Mereology: Mereology is defined as the science or theory of parts, and is used to describe the part-of relation and its properties.

The components ontology is used to represent the components in a building model and their part-whole decomposition in this research. A component is a general concept that encompasses all individuals used to describe the structure of an object. A component is considered to be atomic if it cannot be decomposed into any further parts. Whereas, components can be part of an assembly. However, assemblies can be made up of atomic components or smaller assemblies. Part-whole relationships are of two types, namely, Part-of, and Proper Part-of relations. Part - of is the general relationship that covers all the individuals in this ontology, whereas Proper Part-of restricts this relationship using the *Weak Supplementation Principle*. This principle states that, when an individual has a proper part, it must have another proper part disjoint from the first. That means the individual cannot be distinguished from the sum of its parts. A perfect example is the slab beam aggregation. A slab is the aggregation of individual beams, which means that beams are proper part of the slab. Whereas, the project-site- building-building storey, space hierarchy is simply a Part-of relationship. Moreover, in the case of proper part of relationship, the geometry of the

parent is the resulting sum of the individuals. This is expressed by the following definition.

$$x < y \equiv x \ll y \bigvee x = y(ProperPart - of) \tag{1}$$

Transitivity also holds for Part-of relationship. Transitivity states that when an individual is a proper part of a second individual that is a proper part-of a third individual, then the first is also a proper part of the third (A part of B and B part of C, then A part of C). For example, Building has slabs, slab has DoubleTee, hence building has DoubleTee. This is expressed by the following definition.

$$x \ll y \bigwedge y \ll z \supset x \ll z(Transitive) \tag{2}$$

Transitivity can be used to define assemblies as being assembled from parts. Asymmetry makes it impossible to say that an individual is a proper part of itself. A is a part of B, then B is not a part of A.

$$x \ll y \supset\sim y \ll x(Asymmetry) \tag{3}$$

Overlap and disjointness are defined as sharing a common part or the negation of this as expressed by the following definitions. An individual overlaps another means that either one is a part of the other.

$$x \circ y \equiv \exists z z < x \bigwedge z < y(Overlap) \tag{4}$$

$$x \mid y \equiv\sim x \circ y(Disjointness) \tag{5}$$

According to the weak supplementation principle, when an individual has a proper part then it must have another proper part disjoint from the first, which means an individual cannot be distinguished from the sum of its parts. This is shown by following equation.

$$x \ll y \equiv x < y \bigwedge \sim y < x(ProperPart - of) \tag{6}$$

A good example satisfying this axiom is the Building Element being a proper part of another building element, such as a slab aggregation. Slab's component pieces are assumed to be mutually spatially disjoint, without overlaps. They may overlap the slab. Slabs are a

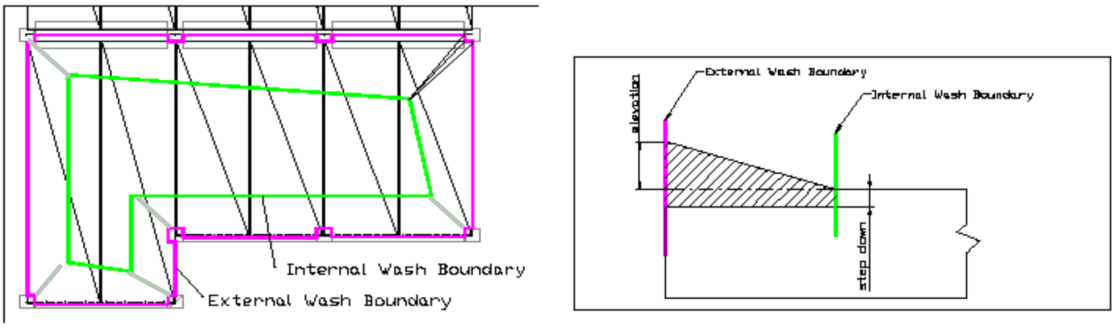

Figure 37: Aggregation of individual components into a slab [34].

composition of individual precast pieces, such as hollow core, DT or solid slabs. The cut shapes of these components fit inside of the slab shape as shown in figure 37. The shape of a slab is defined as a general purpose shape, boundary representation because its top may not be planar because of toppings. The slab shape and its components, when unioned together, should have no spaces between. Thus specific recommendations of shape are defined for each type of embedded beam. We can also have assemblies aggregated into bigger assemblies. Overlapping classifies Proper Part of relationships into two classes here. Those which allow overlapping and those which do not. Example, DT being a proper part of slab, but does not allow overlap. Whereas, reinforcing is a proper part of beam but allows overlapping. Overlap can be checked by taking binary product of two individuals as expressed in following definition.

$$x \cdot y(BinaryProduct) \tag{7}$$

For non overlapping individuals the above equation is invalid. Figure 38 shows the dot product. A beam is resting on a wall, these two individuals are not supposed to overlap. Hence, they cannot have a dot product, and therefore the shared part has to be assigned only to one of the individuals. This necessitates a blockout to be made in the wall such that the beam is supported by the wall as shown in Figure 38.

The binary sum is the individual that encompasses at least one of x and y. The difference x-y is the individual which is a proper part of x but does not share a part with y. The following expressions define this.

$$x + y(Sum) \tag{8}$$

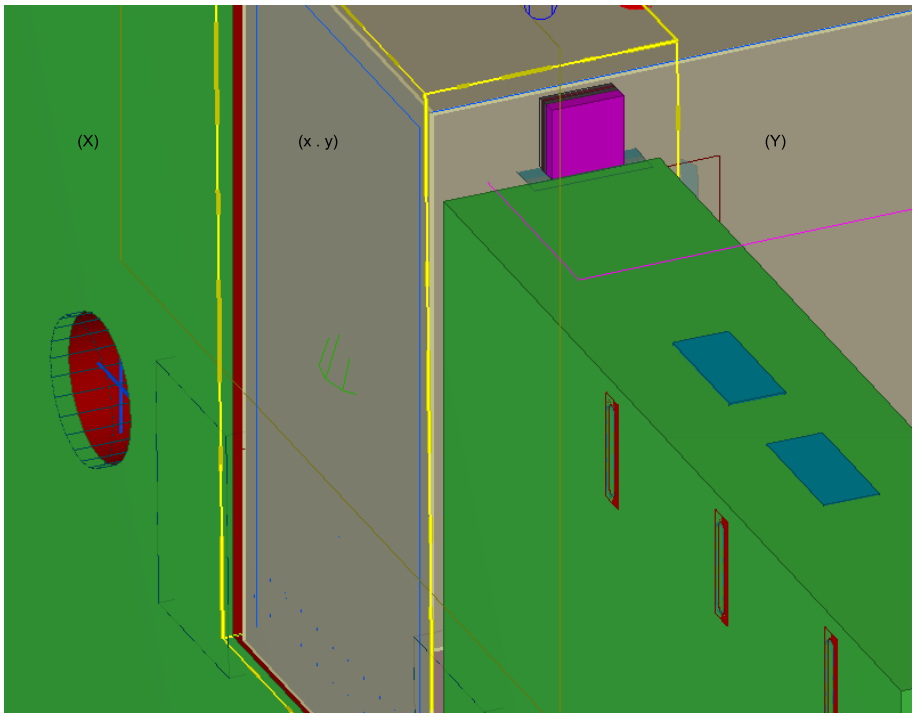

Figure 38: Overlap, binary product, sum and difference in precast components.

$$x - y(Difference) \tag{9}$$

Feature additions and subtractions are examples. Sum provides a boolean addition to a precast piece, such as a corbel. Difference can be used for voids. Discrete accessory proper part of a building element is an example of a proper-part of relationship that allows overlaps. A similar example is voids in a building element.

Slabs contained in a building storey is an example for building elements being a part-of a spatial structure element. If there are building elements and/or other elements directly related to the building (like a curtain wall spanning several stories), they are associated with the building. Similarly, project - site- building - building storey - space hierarchy falls under the relationship of spatial structure element being part-of another spatial structure element.

6.2.3 Connection Ontology: Theory of Topology

Topology describes the behavioral aspects of a system. The theory of topology extends the Component ontology. Along with the part of relationships, this provides the connections

between objects.

Topology: Topology is defined as the science or theory describing the *is-connected-to* relation.

Is-connected-to relationship: It is a reflexive property; any part is always connected to itself. Also it is symmetric. If A is connected to B, then B is also connected to A.

$$x \asymp x(Reflexive) \tag{10}$$

$$x \asymp y \supset y \asymp x(Symmetry) \tag{11}$$

Extending the proper part of relationship, we can say that all individuals that are a proper part of a whole is connected. Or formally, if individual x has a proper part y, then there should be another proper part z to which it is connected. This also holds the Weak Supplementation principle explained in Component Ontology.

$$x][y \equiv\sim x \asymp y(Disconnected) \tag{12}$$

$$x \asymp y \bigwedge \sim x \circ y(ExternalConnection) \tag{13}$$

The *is-connected* to relationship can be restricted as external, if an individual x is connected to y and they do not overlap. The realizing element is the means by which the connection is provided. Since the existence of realizing elements is solely due to the topological configuration of individuals and hence the realizing elements cannot exist on their own. Different types of connections are represented (connection geometry) using points, lines, surfaces, and volume. These are inherited from the geometry ontology. Realizing elements of type reinforcing bar or discrete accessory may be embedded in one of the precast pieces that is part of the connection, or they may be delivered to the site as field hardware. In the former case, the element must also be associated directly with the building element in which it is embedded using an aggregation relationship, in addition to its relationship to the connection as defined here. Specific rules validate the compatibility between the connectors and building element, thereby influencing the validity of the connection. Some examples for the valid connection types in precast pieces are given below:

1. End-to-end connection:

   Figure 39 shows different configurations of end-to-end connection and realization of one of them. Different connection types for end-to-end can be realized using the following: Column base-plate, Socket base, Grout-sleeve base, Bolted, Welded plate, Tube to tube, Grouted sleeve, Welded lap bar, Tube sleeve, Post-tensioned splice, Simple Welded, Doweled, Composite moment, Corbel, Pocket, Sleeve and dowel, Moment-resistant, Architectural bearing, Alignment, Seismic shear plates, Other precast end-to-end connection.

2. End-to-edge connection:

   These include: Column base-plate, Socket base, Grout-sleeve base, Bolted, Welded plate, Tube to tube, Grouted sleeve, Welded lap bar, Tube sleeve, Post-tensioned splice, Simple Welded, Doweled, Hanger, Composite moment, Corbel, Pocket, Sleeve and dowel, Moment-resistant, Architectural bearing, Tie-back, Alignment, Soffit hanger, Masonry tie-back, Seismic shear plates, Other precast point connection.

3. Seam connection:

   These include: Double-tee seam, Wall to Wall doweled, Other precast seam connection.

6.2.4 System Ontology

On top of the component ontology and the connection ontology, a system ontology is defined. This helps to define the different individuals in a system, the connections within the system and also to outside systems etc. We can also have sub-systems. The relationship in-system aggregates individuals into a system. For example, pieces can be aggregated into a precast system. This will also include the embedded individuals etc. A system is made up of individuals, but not every individual is a system.

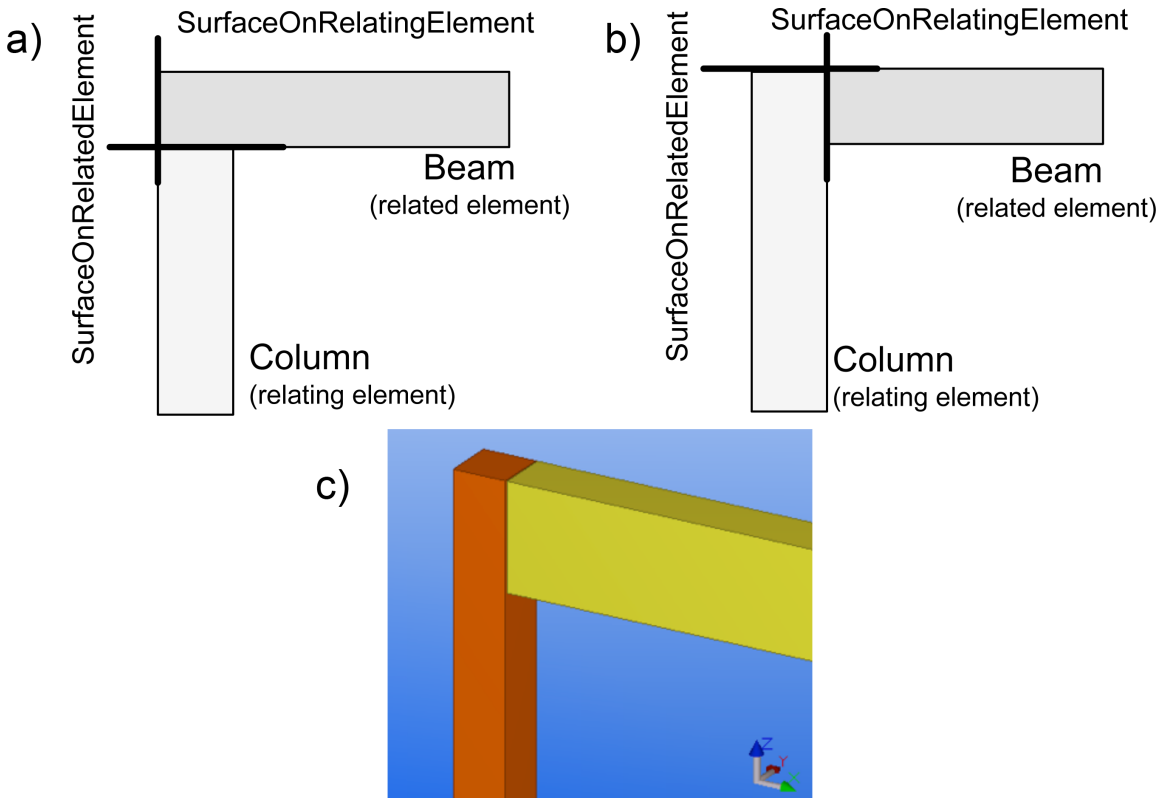

Figure 39: Different configurations for end-to-end connection types. a) and b) shows connection surface on relating and related elements and c) shows realization of a precast piece connection [34].

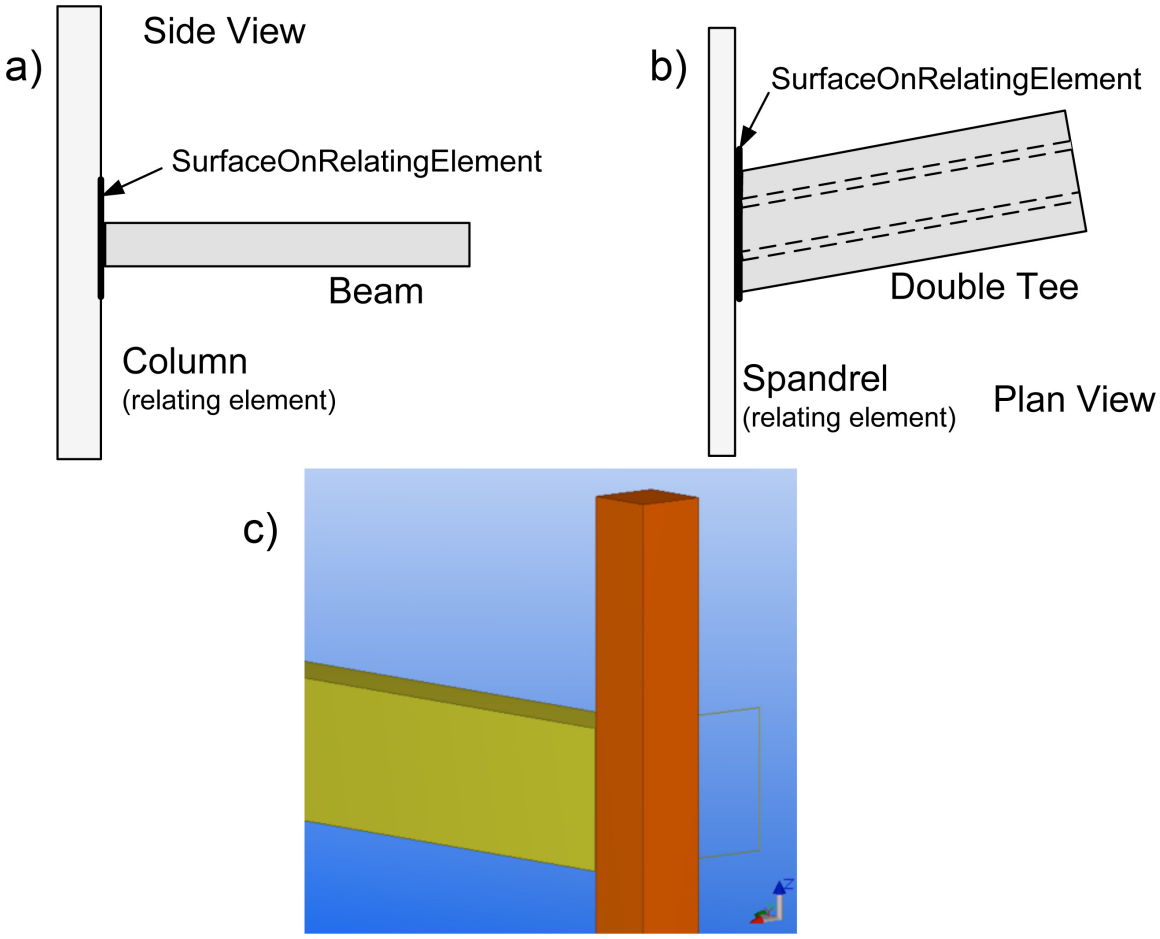

Figure 40: Different configurations for end-to-edge connection types. a) beam connected to a column, b) shows a double tee attached to a spandrel and c) shows realization of a precast piece end-to-edge connection [34].

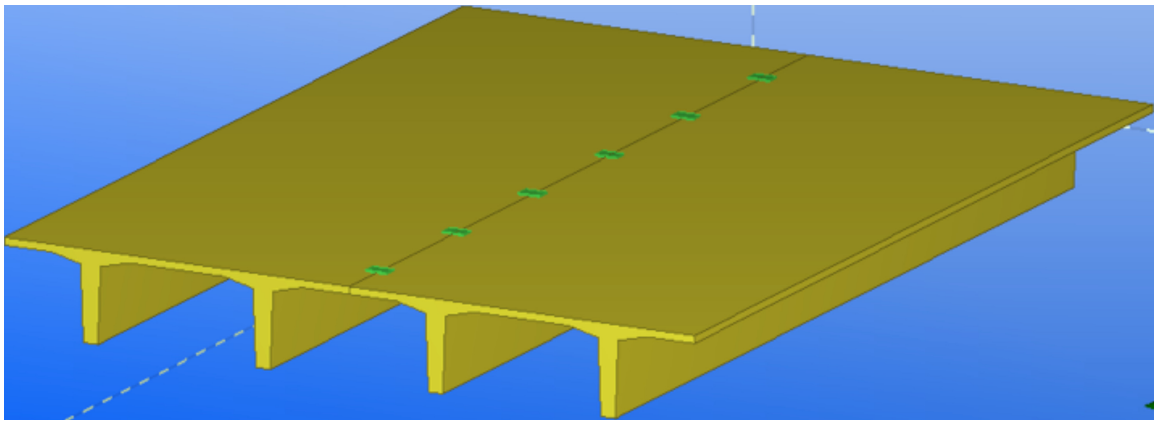

Figure 41: Realization of a seam connection on a precast concrete double tee [34].

6.3 Precast System Ontology

An Application Ontology specifies how the application's functionality is to be implemented and it serves roles similar to ER diagrams, object models, and object patterns. An Application ontology is built on top of engineering ontologies. The Precast System Ontology defines how a precast model should be specified in general, in the form of a set of theories. A precast piece can be modeled using the above defined engineering ontologies which are a part of the application ontology. Depending on the needs we can define a precast piece ontology using component, connections, system, etc. and adding classes for requirements, placement, and geometry. Excerpts of the ontology and graphical illustrations are given throughout this chapter, however for full specification of the precast system ontology refer to Appendix D.

What is an Object? Object attributes general information about the individual. We use the term Object to represent any *physical object* in a model exchange. Figure 42 shows graphically the object classification. *Thing* is the root of all Objects as shown in below definition.

$$PhysicalObject(x) =\succ Object(x) =\succ Thing(x) \tag{14}$$

This definition can be extended to include processes, controls, and actors.

Ontology for Structure of Object The object definition is extended by including structural definitions. The structural ontology is qualified by three relationships *has representation, has material association, and has placement.* An object has material associated with it, however the material requirement is extended and defined in the Requirements Ontology. Every individual has a placement relationship and can be realized by three different mechanisms, namely, absolute placement, placement relative to a grid, and placement relative to another individual. Geometry is an area which has been studied in depth over the years [103]. For purposes of this research we assume that geometry requirements can be as follows

- B-rep Geometry
- Extruded Geometry
- Arbitrary Profiles

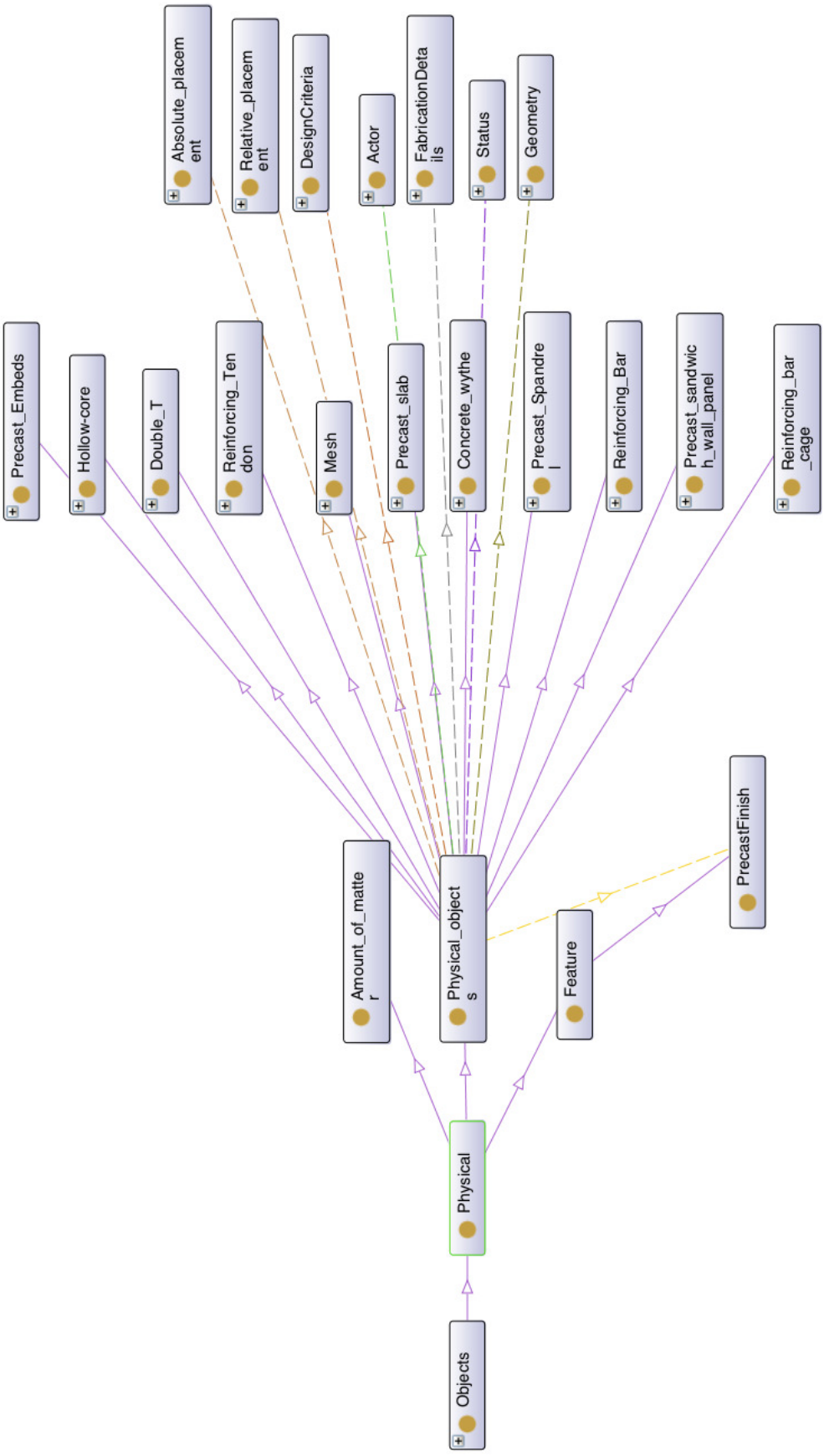

Figure 42: Object classification hierarchy.

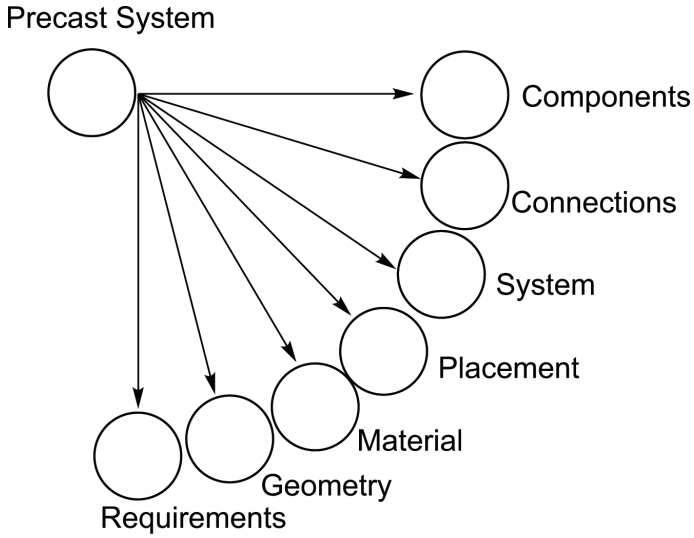

Figure 43: Structure of the *Precast System Ontology* built from separate Engineering Ontologies.

Figure 44 shows graphically the structure of geometry ontology. An additional option is to directly access the shape as provided in a library catalog.

Type-Instance Ontology: Types are defined as a rigid property that has *identity*. This definition is used to differentiate abstract types from quasi types. The idea of quasi-types is based on the work by Guarnio et. al. [56]. Abstract types are used as a means to categorize, for example beams and columns as a building element, where building element is an abstract type. However, quasi types are those defined for organizational purposes by grouping entities based on useful combinations. For example, a piece mark is an example of a quasi type. If a type is defined as a *Class*, then a class is a subclass of another class if all instances of the subclass are also instances of the superclass. For example, all beams are a type of building element, if beam class is defined as a subclass of building element. This is enforced by the following definition.

$$subclsOf(x,y) \prec\!=\!\succ Cls(x) \bigwedge Cls(y) \bigwedge \forall(instanceOf(inst,x) =\!\succ instanceOf(inst,y)) \tag{15}$$

Any individual from the component ontology can be elevated to the level of type. Instances are related using the Type-of relationship. The type can be an atomic component or an assembly. Types can be created from different levels, for example an atomic individual

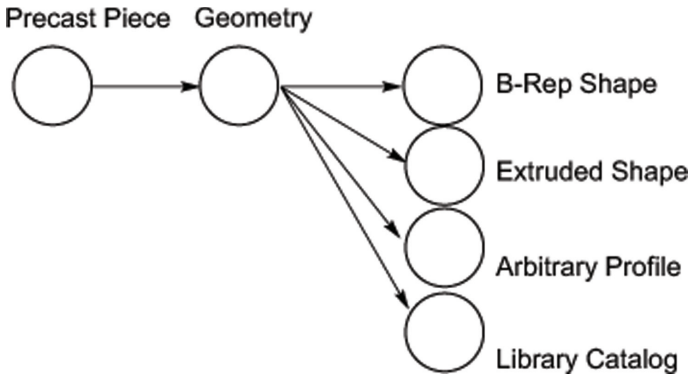

Figure 44: Precast piece linked to the geometry ontology

can be assigned as a type and instances made out of it. Or an aggregation of individuals together can be assigned as a type. Or even a complete assembly with connections etc., can be made into a type. Usually the geometry is attached at the type level and is inherited by the instances. Only special modifications such as additions or subtractions of features is done at instance level.

Requirements Ontology: The Requirements ontology is influenced by the ontology for requirements [19] or quality of objects [86]. The requirements ontology contains main concepts needed for the representation of the function and behavior of individuals. It is important to attach the requirements to the systems and pieces. Property sets are an important notion in IFC data schema, which can be used for specifying requirements. Property sets can also be in multiple levels. For example the requirements for a precast piece can be decomposed into requirements related to performance, design criteria, delivery methods, etc. Classification of requirements are given in [99] on the basis of cost, functional, safety, technological, and ergonomical. In the case of precast systems, requirements should be differentiated on the basis of *as-fabricated* and *as-installed* as well.

6.3.1 Ontology Implementation using Protege

A model is developed for testing the semantics of precast system. The entities required to define this model view are captured from the ontological definitions and defined in Protege as classes. The approach of grouping sub-classes into high level concepts is followed in this research. For example, all the entities required for representing the geometry are classified

under one class, all the entities required for placement separate, entities for components are separately defined etc. Figure 45 shows a tree structure of the entities created in Protege. Similar to this, the relationships (and inverses) are created and defined. There is a major difference in how relationships are treated when compared to IFC. For this model the relationships are considered as first class objects unless otherwise defined. Hence, it is important to define the classes and relationships in a strict way by implementing the semantic meanings associated. Table 8 shows an excerpt of the Precast System Ontology developed in Protege and figure 46 the corresponding definition in OWL.

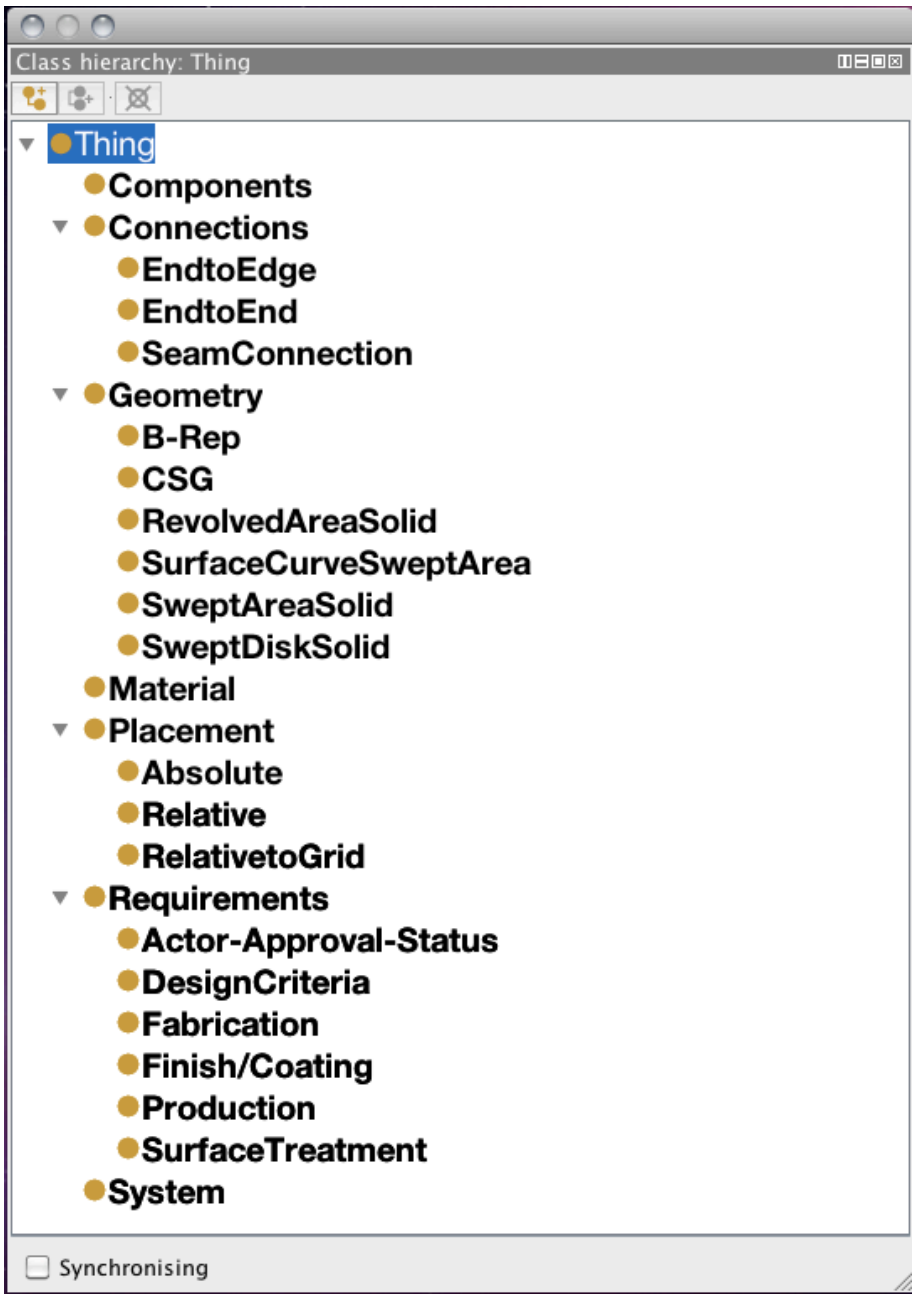

Figure 45: Hierarchical view of entities created in Protege.

Table 8: A subset of the OWL representation of the Precast System Ontology.

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://dcom.arch.gatech.edu/ontologies/2010/
     IFCOntology.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     ontologyIRI="http://dcom.arch.gatech.edu/ontologies/2010/
     IFCOntology.owl">
  ..... ...... (break)
    <Declaration>
        <Class IRI="#Extrusion"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Geometry"/>
    </Declaration>
    <Declaration>
        <Class IRI="#IfcBuildingElement"/>
    </Declaration>
    <Declaration>
        <Class IRI="#IfcBuildingElementPart"/>
    </Declaration>
    <Declaration>
        <Class IRI="#IfcDiscreteAccessory"/>
    </Declaration>
    .... .....    (break)
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
        <IRI>#IfcObjectPlacement</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">Abstract
        supertype for the special types defining the object
        coordinate system. The IfcObjectPlacement has to
        be provided for each product that has a shape
        representation.</Literal>
    </AnnotationAssertion>
```

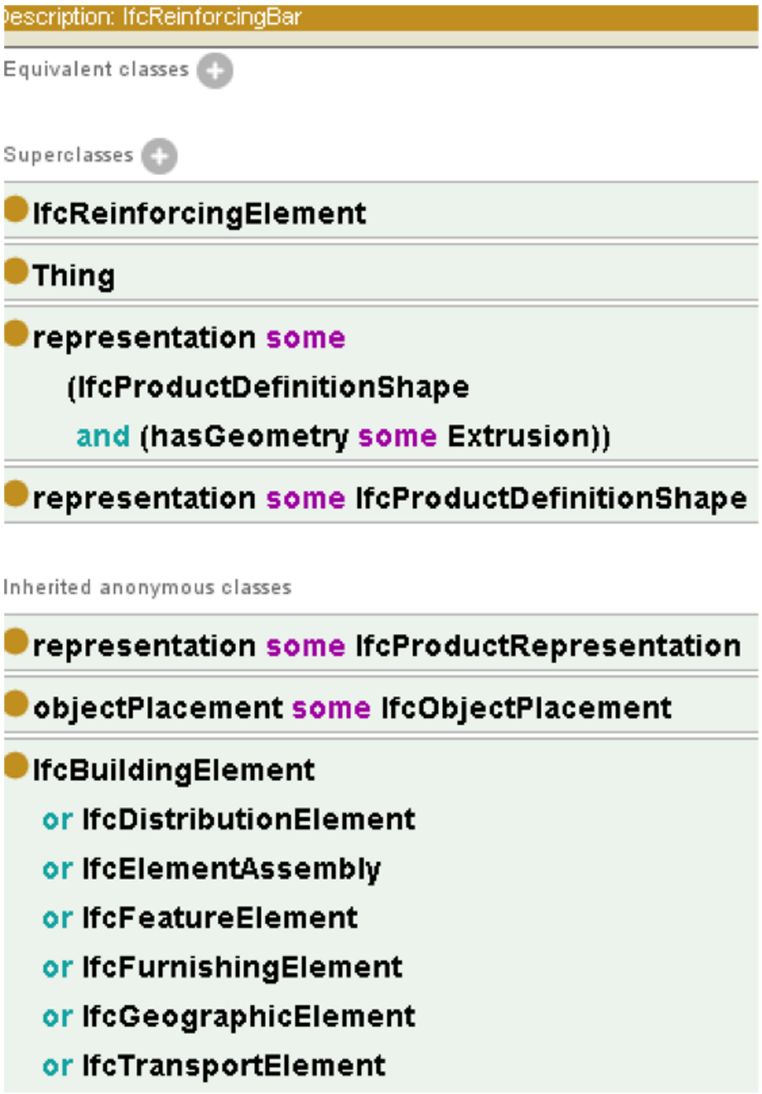

Figure 46: Definition of Rebar class in protege.

CHAPTER VII

SEMANTIC EXCHANGE MODULES

The idea of Semantic Exchange Modules (SEM) is introduced as a means of modularizing model view development and also for improving reusability. This chapter introduces the theoretical foundations on which SEMs are based followed by a mapping of ontological definitions into IFC schema to develop the SEMs.

7.1 Definition

An SEM is a structured, modular subset of the objects and relationships required in each one of multiple BIM exchange model definitions. Its raison d'être is to enable BIM software companies to code import and export functions modularly, such that a function written to export or import model objects according to any given SEM can be tested and certified once, and then re-used to fulfill multiple exchange model exports/imports without modification. An SEM can be expressed conveniently as a binding to a set of IFC entities attributes, relations, and functions. The scope of SEMs must be determined in consultation with software tool developers, since they must map not only to an Exchange Model, but also to the internal object schema of the tool.

7.2 Why SEM are needed?

Semantics in the areas of engineering and design are particular, in the sense that they define a mixture of partial specifications of reality, the expected function and behavior of that reality, and the reality of physical systems. Semantics regarding the different levels of realization seem to be needed to distinguish between definitions and objects within a domain. IFC provides a schema to define instances of specifications of both building designs and real buildings. Buildings are described by terms that vary in their generality and like other taxonomies of engineering and design, with varied levels of realization. Buildings

are made up of many different systems which have their own entities, as well as shared ones. This implies that there is more than one way of representing the information to be exchanged. The notion of human cognition versus machine readability is an important one. The human mind is able to think at different levels of aggregation at different times, whereas software applications need clear definition of the semantics. IFC lacks the formal semantics to relate the entities and relationships to reasoning mechanisms. There is no firm class or property hierarchy (except for project-site-building) as most classes and properties are direct subclasses of the top-level entity. Further, several relationships take *IfcRoot* as their domain or range, which is too broad and does not restrict the usage of the relationship. Some of the implicit semantics are in the IFC documentation whereas some semantics are left to the users for future work. To overcome this situation, the level of commitment in IFC will need to be raised and we recommend a formal definition for IFC extensions, by identifying the underlying semantics used to define the meaning of IFC entities, relations and attributes and constructs, as these are the basis for the IFC structure. Identifying and developing such semantics for exchange modules is explained in this chapter.

7.3 Requirements for SEMs

Various requirements for Semantic Exchange Modules (SEMs) are discussed in the following sections.

7.3.1 Expressiveness

Expressiveness: Expressiveness is the extent of possibilities or representation provided by the exchange modules.

Expressiveness is synonymous to the functionalities provided by a system. In this case, we are concerned by the ability to express the requirements specifications in to a model view using semantic exchange modules. With increase in expressiveness of an SEM, the complexity of system also increases and hence there is a trade off. Moreover, a highly expressive system lacks computability/reasoning. IFC is rich in expressiveness and SEM is proposed to be restrictive as compared to IFC by providing additional layer of specificity.

7.3.2 Ease of use

Ease of use: Ease of use is the ease with which people of various backgrounds and qualifications can learn to use SEM and apply them to solve problems. It involves exchange specifications, model view definitions, and implementations.

In terms of ease of use, SEM is positioned as an intermediate layer to natural language (very easy) and high level programming languages (too complex) and hence domain experts as well as programmers can understand model views represented in terms of SEM.

7.3.3 Composability

Composability: Composability is the ease of combining entities together into a module.

From a semantic point of view, there should be no broken links or references and semantic relationships between terms should be explicitly defined. Specifically, a SEM should allow bindings with other SEMs, without editing their interface, or adding or subtracting of references external to the SEM.

7.3.4 Reusability of SEMS and MVDs

Reusability of SEMS and MVDs: Reusability is the ability of entities to be used for the construction of many different applications.

An important requirement, which was identified during the current model view work, is the need to avoid redundancy and rework in terms of development and testing of model views. Hence, SEM should be generated following strict/formal rules so that they are testable and standalone. For new MVD development, these should be in a plug-and-play form. Retesting, which is expensive and time consuming needs to be avoided. Such modular SEMs can be plugged in wherever there is a requirement. The implication is that an SEM should be general enough to support all its functionalities, or in other words, it should go way beyond the functionalities that were initially targeted. Otherwise it cannot be considered fully reusable.

7.3.5 Correctness

Correctness: Correctness is the ability of entities to follow the correct specification.

Correctness is the prime qualifier. It ensures that the SEM satisfies or represents what the specification is. Methods of correctness are conditional and are based on testing.

7.3.6 Robustness

Robustness: Robustness is the ability to react appropriately to abnormal conditions.

Robustness extends correctness. Correctness is tested based on a set of specifications whereas robustness is tested in the case of what happens outside that specification. Some of these as and when identified will be added to the set of specifications. Hence, the conditions for correctness will grow as and when new requirements are identified.

7.3.7 Extensibility

Extensibility: Extensibility is the ease of adapting modules to changes in specification.

We need extensibility because IFC is an extensible schema. New requirements for various domains are identified and proposed in due course. Such new requirements should not invalidate existing specifications. It should follow an open-closed principle. (Open for extension, but closed for change). The main factors affecting extendibility are simplicity and decentralization. Simple hierarchies are always easier to adapt changes and the more autonomous the modules, the higher the likelihood of introduction new requirements as new modules. Upward compatibility of schema is major issue facing IFC extensions.

7.3.8 Traceability

Traceability: Traceability is defined as the ease of analyzing a model view and mapping them back to exchange requirements.

Traceability is synonymous to reverse engineering. Model views represent different levels of

detail; hence the new methodology should contribute to a better under- standing of model views by providing a concise and object oriented view of the exchange. It should be possible to decompose the view into several modular objects that are more manageable and testable. Traceability is a very important feature in the development process to trace the model view back to the exchange requirements. This can also be called as verifiability and goes back to maintainability of model views.

7.3.9 Timeliness

Timeliness: The ability to develop and implement the exchange requirements in a timely manner.

The current model view development lifecycle of 2-3 years should be reduced to a more practical 6-8 months. This will help to introduce implementations in a more timely manner.

7.4 Structure of Semantic Exchange Modules

7.4.1 Desired Features

The choice of a proper module structure is the key to achieving the aims of reusability and extendibility. The issues of decomposition (top-down) and composition (bottom-up) needs to be addressed. The *Open Closed Principle* is another important desirable feature. The SEM structure developed should support querying of content-based data from a product model or from a model server. The current terminological and semantic ambiguities need to be removed by the formal structure so that it minimizes semantic mismatch during querying for various applications. Accomplishing this task requires the SEM definitions and constraints (business rules) to be represented rigorously. A more effective translation and transparency of the user needs (Exchange Requirements) into the design of MVDs is required.

7.4.2 Two Dimensional Graph Structure

The notion of a SEM is that it is a subset of a product model schema that can be used to create various, higher-level, model view definitions (MVD). The high-level SEMs are called the Object SEMs and they are intermediate to natural language and IFC entities such

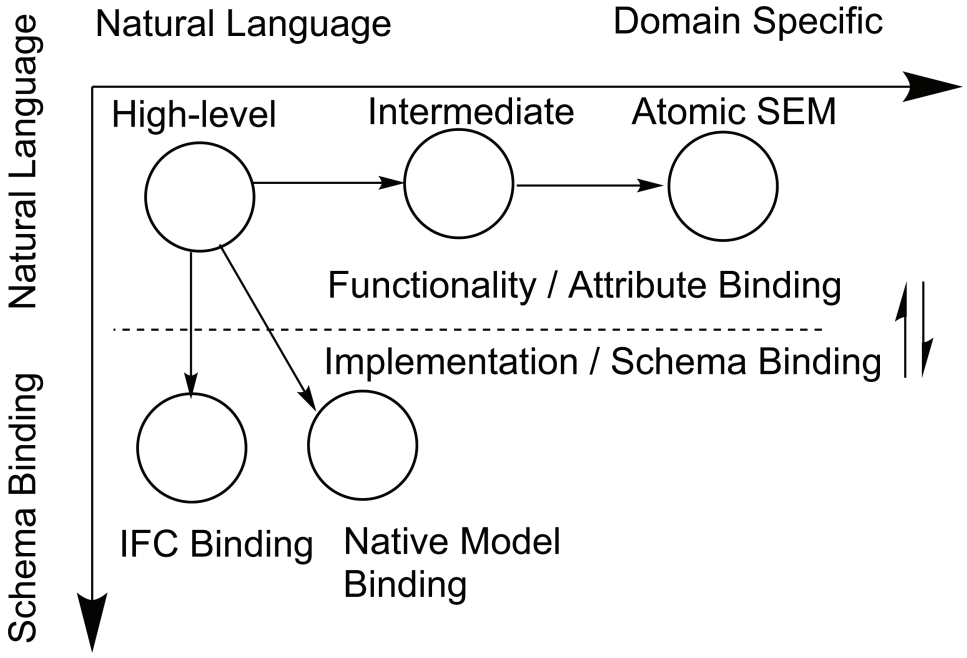

Figure 47: Mapping of Semantic Exchange Modules to the IFC schema and native models.

that a domain knowledge expert can easily draw up requirements based on these high-level concepts. The second level consists of Median SEMs. As the name suggests they have different implementations according to the context. Hence, these are context specific and often require low-level concepts, i.e. Atomic SEMs to be tied to them to make full meaning. The Atomic SEMs have a definite structure and behavior. The main criterion in defining an SEM as Atomic is that it cannot be allowed to show polymorphism (in object-oriented terminology). A SEM graph, usually has two dimensions as shown in Figure 47. The first dimension is the classification hierarchy of different entities involved and their relationships. The second dimension involves the implementation of each of these nodes in the graph by mapping it to a schema (IFC and native). The branches of this dimension represent the data access paths. Therefore, a SEM has:

1. a definite mapping to a schema,

2. mappings to a native model (when fully defined),

3. methods to map between the two bindings,

4. data access paths and

5. belongs to a specific classification hierarchy.

Such a structure with the above mentioned branches makes SEMs executable.The main criteria to be satisfied for creating such executable modules is composability as explained in the previous section (and Figure 48). *Can we specify model views by combining SEMs with each other?* If the SEMs are sufficiently autonomous/independent then we can assume this is possible. For example, if we have such exchange modules for B-Rep geometry, placement, material, features, etc., then it should be possible to compose them together to satisfy a precast model view. This is analogous to building a system from standard predesigned elements. Composability can be seen as a bottom-up approach and this is in stark contradiction of how IFC is designed as a top-down structure.

7.4.3 Independent EXPRESS Subschema

Another major criterion of SEM is that they need to be stand-alone and testable from a completeness point of view. SEMs should be composable into a complete subschema that has no broken links or references. This is synonymous to decomposing a complex EXPRESS schema (or a model view) into a small number of less complex, valid sub modules, connected by a simple structure. This should be independent enough to allow development to be done separately using these sub-modules.
Two criteria:

1. The dependencies between modules should be kept to a minimum.

2. The dependencies should be explicitly defined.

An example is the spatial configuration SEM, as shown in Figure 49. The project-site-building-building storey-space can be combined into a module and the dependency is the spatial containment relationship, which is used to assign an object into this configuration. In other words how other modules make use of this module should be clearly stated. A trade-off is that composing entities into modular unit and decomposability of Express schema are contradictory to the deep inheritance hierarchy. Both are part of the requirements

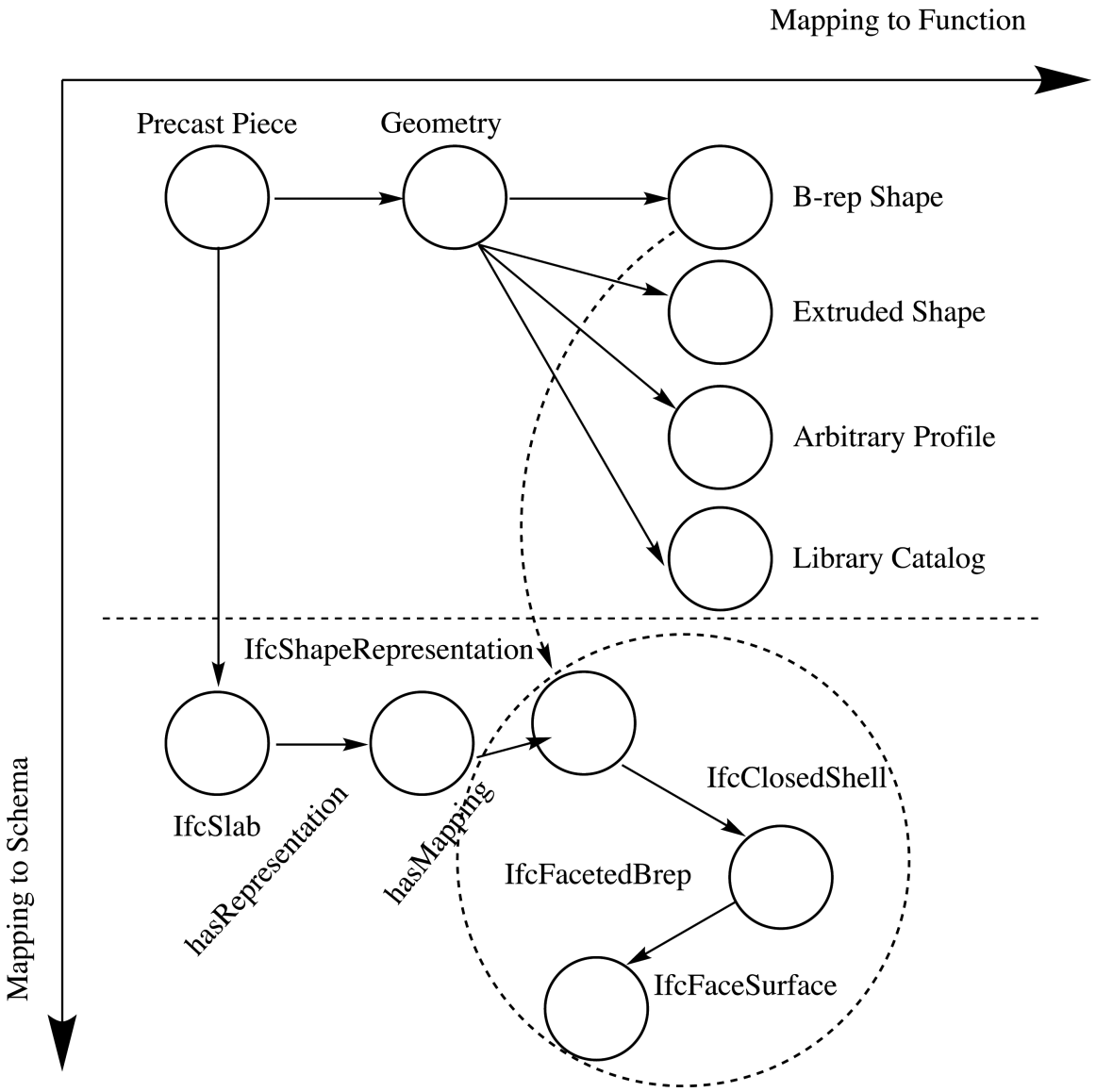

Figure 48: Graph structure for geometry SEM illustrating the two dimensions i) x-axis shows functionality mapping and ii) y-axis shows mapping to schema.

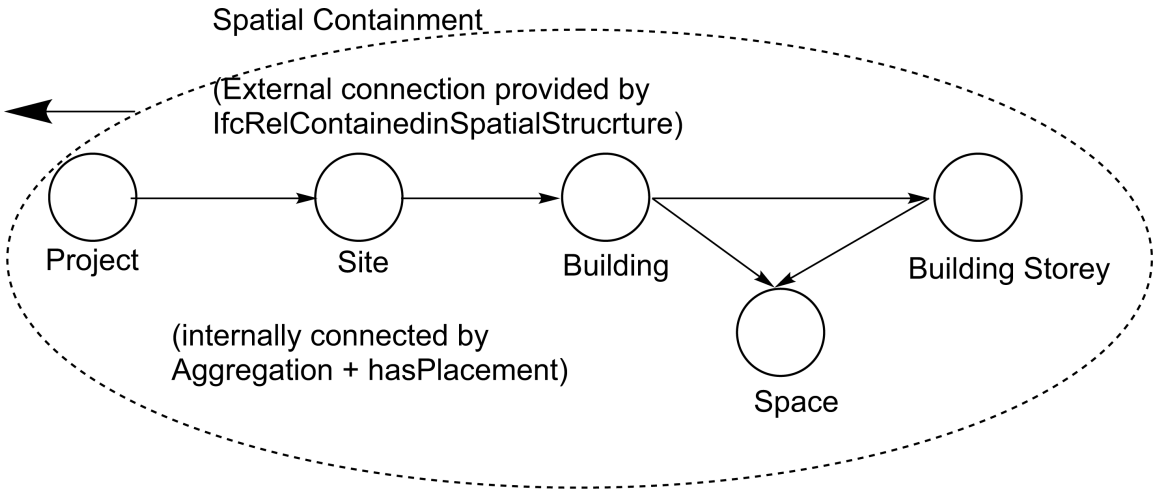

Figure 49: Semantic Exchange Module for Spatial Containment.

for a modular method and there should be a balance with the top-down and bottom-up approaches.

7.4.4 Object-Oriented Principles

Open-closed principle: Modules should be open for extension and closed for change. A module is said to be open if it is still available for extension. For example, it should be possible to extend its use to other domains by adding external entities. A module is said to be closed, if internally the entities and relationships between them are well defined and need not be changed for different contextual use. If a SEM violates this principle then it is an indication that the module either needs to either broken down into more than one smaller modules, or expanded to include more entities. This could be a good guideline in drawing the boundaries of SEMs. Some suggested guidelines:

1. If IFC entity structures are always composed in a given way, they should be combined in an SEM.

2. Conversely, SEMs should have boundaries corresponding to variations in binding structures.

3. If a structure is optional and not always used, but always has the same structure, it should be included in a single SEM, to aid simplification and parsimony. (Example is

the spatial configuration hierarchy.)

These variations apply to object structures (Entity and Type). Attribute-value dependencies across SEMs are often necessary and need to be documented, but do not require partitioning.

Weak coupling: The interfaces between modules should be as minimum as possible. This allows modular continuity and protection. A system can be said to be continuous if a small change in the specification triggers a change in the least number of modules. Protection is useful if one of the modules needs to be redefined, then the change is restricted to only that module or to the least number of neighboring modules.

Design patterns: Following established OO design patterns [44] help in reusability.

The development of SEMs should follow the criteria set by Object Oriented principles. Some of the metrics can be as follows [29]:

1. **Weighted Methods per SEM (WMS):**

   This metric is originally defined in literature as the summation of the complexities of the methods in a class. In this research, this can be translated as the weighted methods per SEM. If all complexities are considered to be unity, then the value of this metric will be number of methods. This metric is related to the complexity of a thing, since number of methods is a predictor of how much time and effort is required to develop and maintain the SEM.

2. **Depth of Inheritance Tree (DIT):**

   Depth of Inheritance of a SEM is the maximum length from the node to the root of the tree. DIT metric relates to the scope of properties as it is a measure of how many parent entities can potentially affect this entity. The deeper the entity is in the hierarchy, the greater the number of methods it is likely to inherit. Therefore a longer DIT metric implies a more complex behavior. However, deeper the tree, there is also more potential to reuse the methods.

3. **Number of Children (NOC):**

   This metric corresponds to the number of immediate entities subordinated to an entity

in the hierarchy. This metric also corresponds to the scope of properties as it shows how many children are going to inherit the properties of the parent. Greater the number of children, greater the reuse.

4. **Coupling between SEMs (CBS):**

   This metric is derived from coupling between the object classes metric [29], which states that the coupling between objects is the count of the number of other classes to which it is coupled. In this case, it becomes the count of number of other SEMs it is coupled to. Excessive coupling between SEMs is detrimental to modular design and prevents reuse. The larger the number of couples, higher the sensitivity to changes in other parts making it difficult to maintain.

7.5 Semantic Mapping to IFC Schema

The Precast System Ontology defined in Chapter 6 included definitions for components, connections, system, geometry, material, placement, and requirements. Semantic Exchange Module structures are developed based on the Precast System Ontology and mapped to the IFC data Schema. Excerpts of important SEMs are provided as follows.

The component ontology provided Part-of and Proper Part-of relationships and definitions. It was shown that to qualify for a Proper Part-of relationship, the weak supplementation principle needs to be satisfied (Refer Chapter 6). Based on this principle we can say that for a Proper Part-of relation the geometry of the parent will be the combined geometry of its parts. The PCI team developed IFC bindings for Building Element aggregation and it was seen to match the ontology definitions. Building elements aggregated into an assembly of building elements and assemblies aggregated into higher assemblies qualify for this relationship. Figures 50 and 51 illustrate this relationship in mapping to IFC schema [34]. Slabs are a composition of individual precast pieces, such as hollow core, DT or solid slabs. The cut shapes of these components fit on the inside of the slab shape. The shape of a slab is defined as a general purpose shape (boundary representation), because the top of the slab may not be planar owing to toppings. Carry should be made to ensure that the slab shape and its components, when unioned together, have no spaces between. In Figure

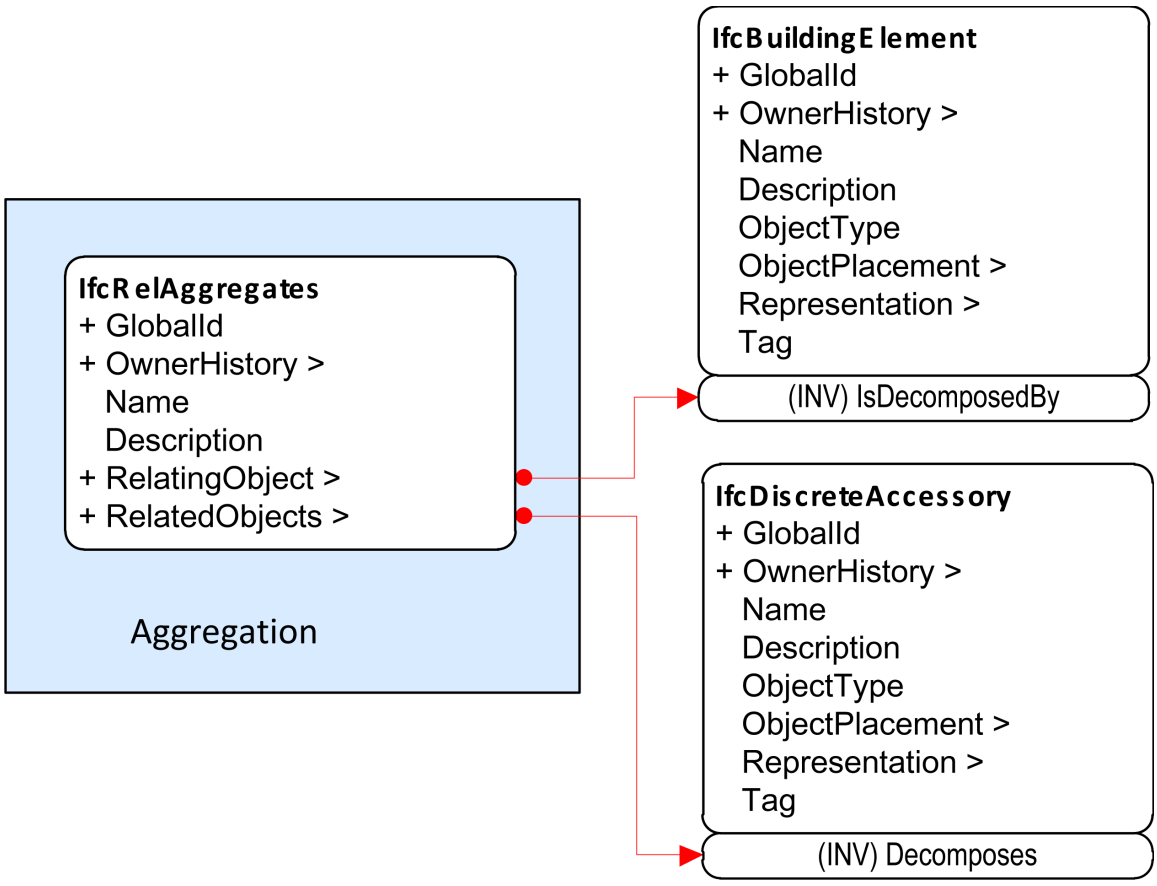

Figure 50: Building element being a *Proper Part-of* another building element [34].

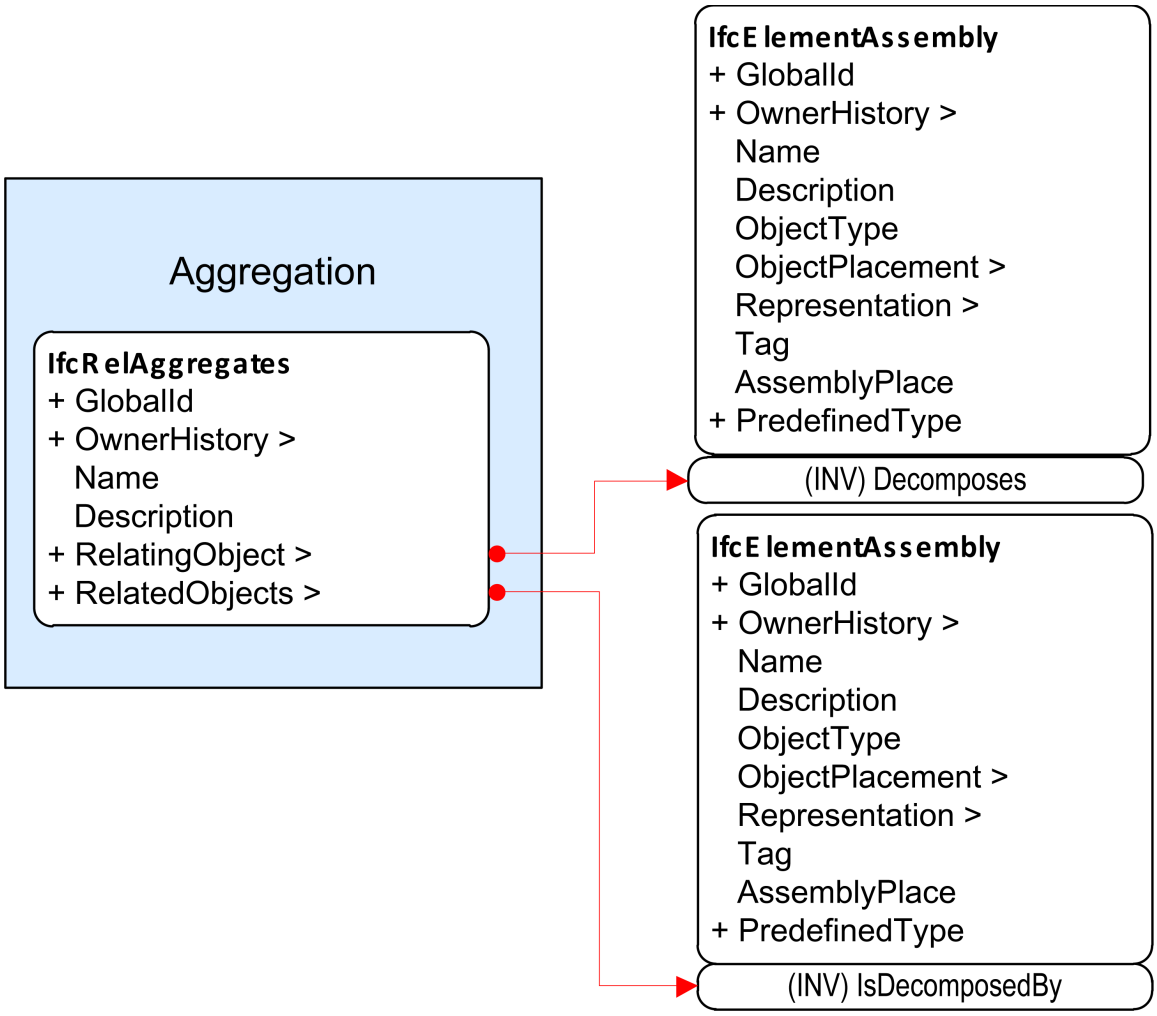

Figure 51: Assemblies being aggregated into higher level assemblies [34].

50 the *RelatingObject* refers to a slab entity with geometry, material, possibly embeds that are within the slab itself, but not in its other components. The *RelatedObjects* references each of the component beams in this slab. Slabs component pieces are assumed to be mutually spatially disjoint, without overlaps. They may overlap the slab. An example for a *Part-of* relationship is the building element contained in a spatial structure element. The differentiating factor between the Proper Part-of and Part-of relationships is that the geometry of a spatial structure container cannot be deducted from the aggregation of the building elements in the space. This is a very important consideration that needs to be taken into account for calculating spaces. Figure 49 illustrated how a building element can be aggregated in a spatial container.

The SEM structure for placement is shown in Figure 52. This SEM provides three different options for placement and each of which will have its own mapping to IFC schema as shown in Figure 53. Similarly material data can also be attached to a building element using the material SEM as shown in Figure 56. The type-instance ontology defined cannot be directly mapped to the IFC schema in the present form. According to the ontological definitions, any object can be elevated to the level of a type, whether it is an atomic piece or an assembly or an assembly of assemblies. Such a flexible typing mechanism is not available in IFC schema at the time of writing. However, if an *IfcTypeAssembly* is introduced in the future release this can be solved. The connection ontology extended the component ontology and provided the *is-connected-to* relationship. The realizing element is the means by which the connection is provided. The realizing element must be a one of *IfcDiscreteAccessory, IfcReinforcingBar, etc.*. To illustrate the implementation of component and connection ontologies, let us look at a scenario where a precast beam is connected to a precast column. There is also a feature addition to the column in the form of a corbel. Figure 55 shows the representation of the same in a BIM modeling tool and the realization of the same in terms of SEMs. This system can be assigned as a Precast System based on the system ontology. Based on the definitions, the Precast System under consideration is comprised of the column, beam, the corbel, as well as the bearing plate. Even though the bearing plate is a steel piece, it is still attached to the Precast System based on the system theory. The

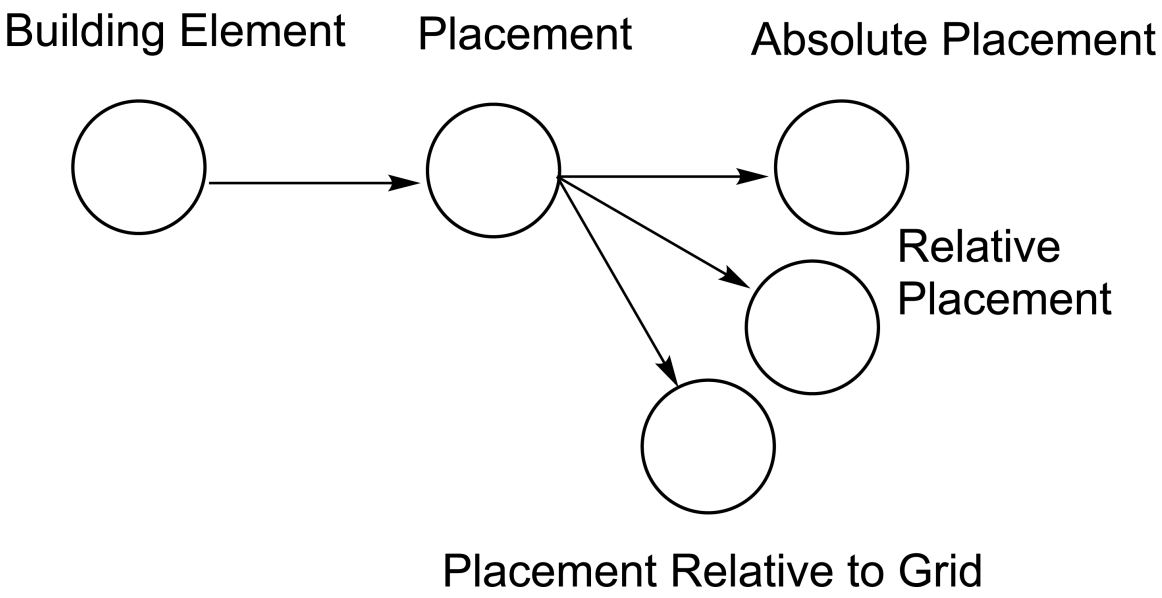

Figure 52: Illustration of SEM structure for Placement of Precast pieces and systems.

different property sets required for the Precast System can be attached to either the pieces, assembly or the system using various property sets. These are optional and are defined based on requirements. The requirements ontology approach allows to attach different functional requirements to the same model, without creating different models. For example, the as-installed and as-fabricated functional requirements can be linked and necessitate two different shape requirements.

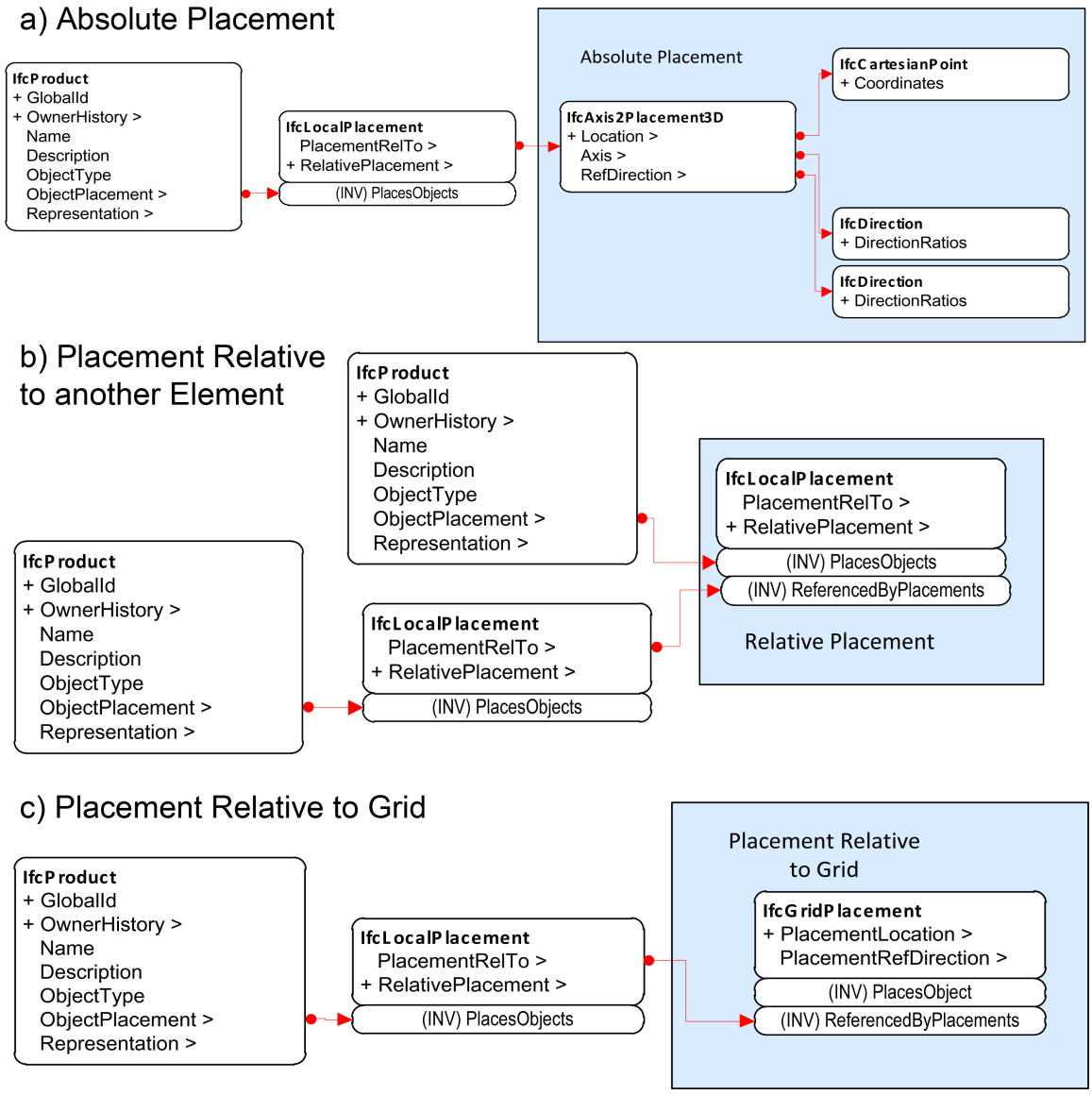

Figure 53: IFC schema mapping for different types of placement for precast piece, a) Absolute Placement, b) Placement Relative to another Element, c) Placement Relative to grid [34].

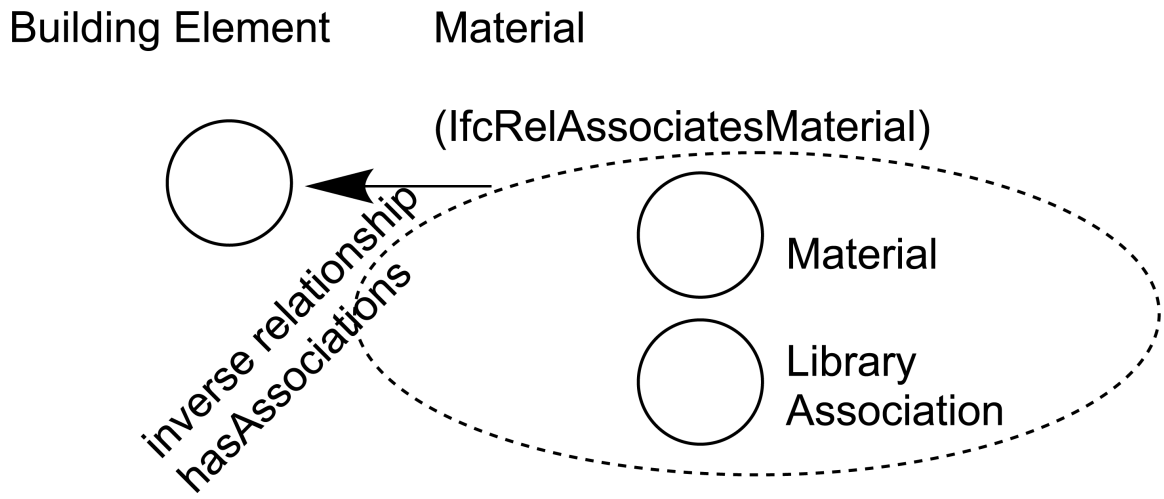

Figure 54: Different ways to attach a material to a precast piece, i) using IFC Material or, ii) using an external library

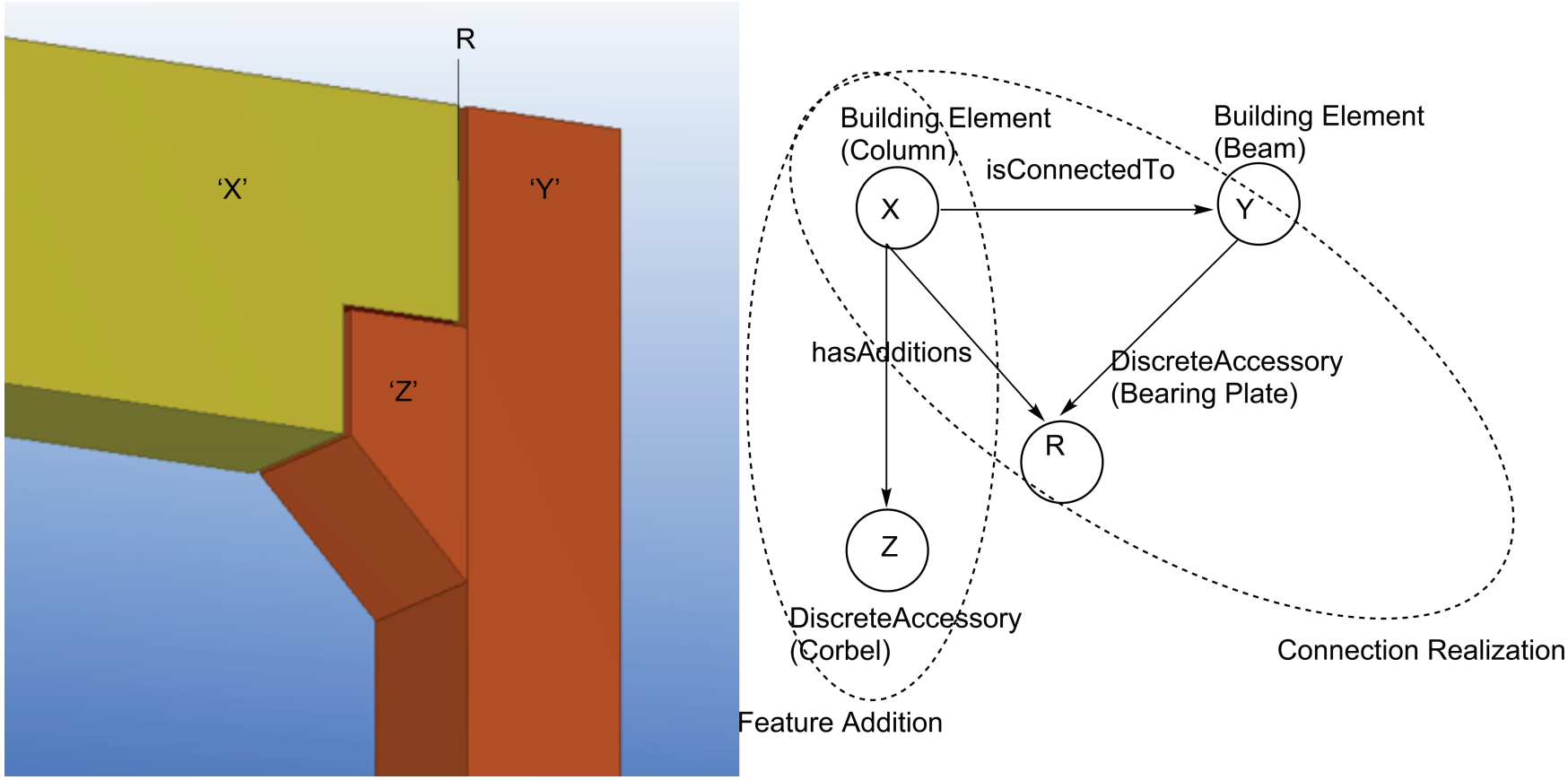

Figure 55: A Precast System scenario showing a beam to column connection and supported by a corbel feature addition

7.6 Results: SEM Library for Precast Objects, Attributes, and Relations

This chapter presented the mechanisms to define a SEM structure and also the mapping to the IFC data schema. This only satisfies one branch of the SEM structure, the other branch being the mapping to the native model schema as shown in Figure 57. The mapping to the native model schema is also required to realize the full potential of the SEM notion. However, the mapping to native model schemas can be implemented only with the support of software vendors. This is an existing limitation of this research and needs to be taken up in the future work. The implementation of mapping to native model schemas can potentially raise questions on the boundaries on which SEMs are modularized necessitating fine tuning. This concern is acknowledged and taken into consideration in the validation plans. However, the main contribution of this research is the new approach of defining the exchange specification based on engineering ontologies and SEMs.

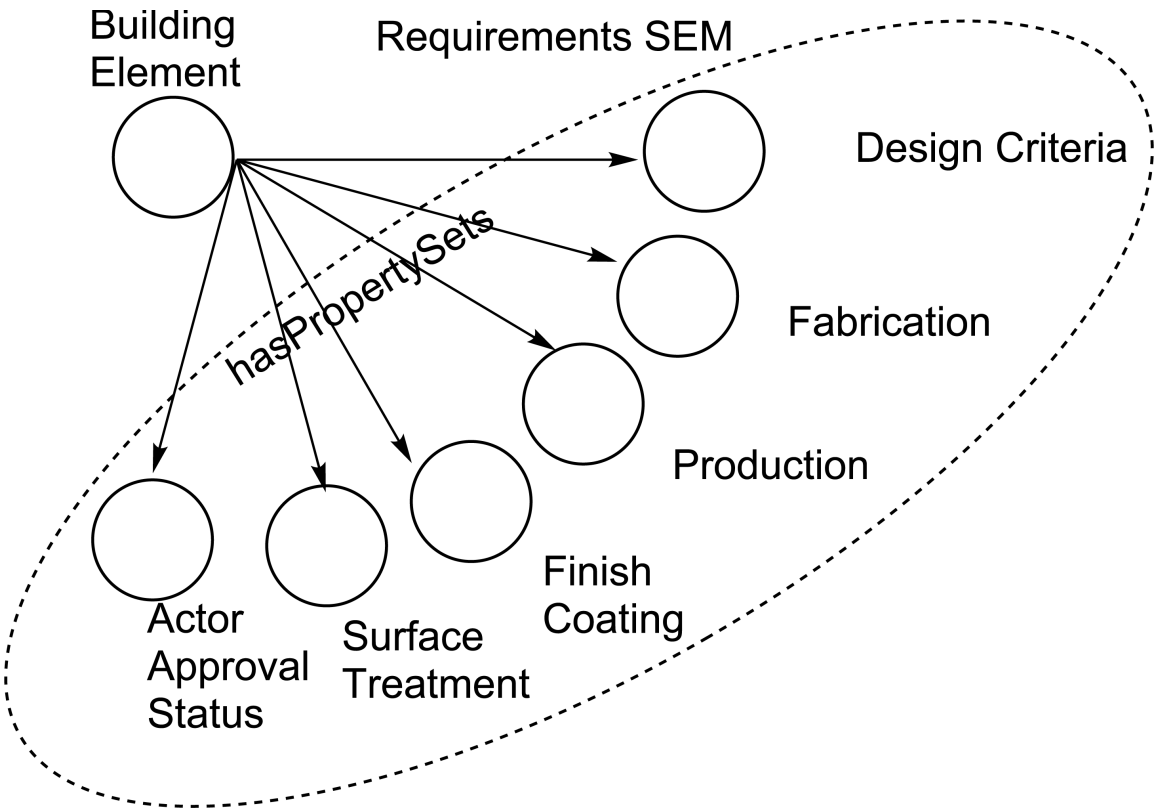

Figure 56: A requirement ontology approach to attach different property sets to the Precast System.

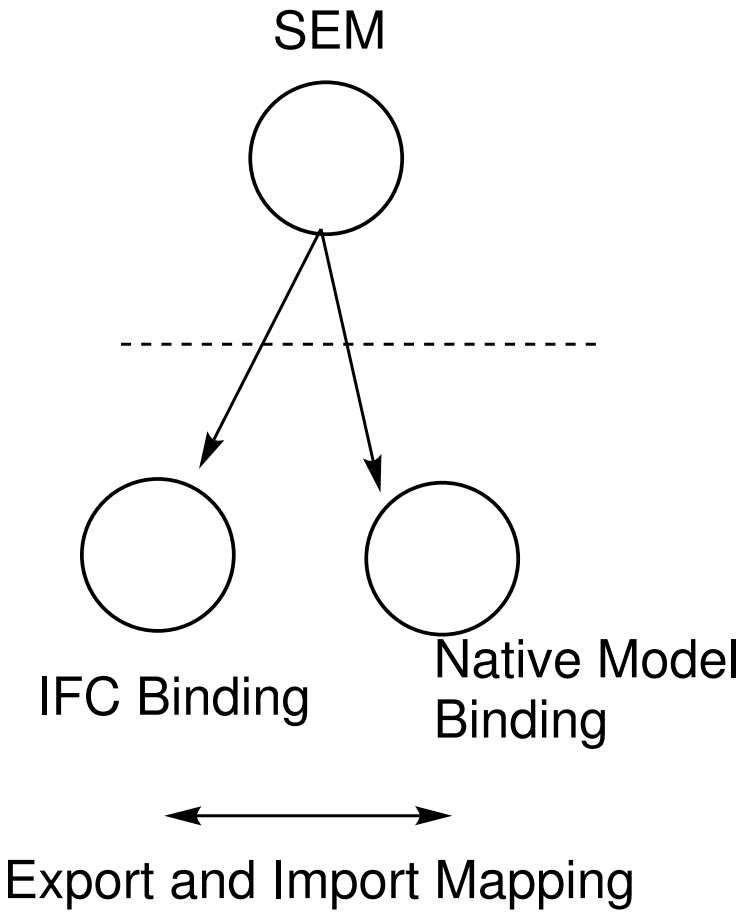

Figure 57: Two branches of a SEM structure, i) mapping to the IFC data schema, and ii) mapping to native model schemas. This research is focused on implementing the mapping to the IFC schema alone.

CHAPTER VIII

VALIDATION AND VERIFICATION

This chapter introduces the metrics for validation and verification of this research. Reasoning engines are used to verify the ontology development work, model view schema checking is performed using EXPRESS parsers, and long term validation plans involving several BIM software developers are discussed. A model view developer plugin is built in C# to illustrate the modularization of model views using SEMs and Ontology.

8.1 Metrics for Validation

The semantic analysis of the IFC schema and definitions for SEM have given insights to the requirements for verification. This is presented in the form of an evaluation metrics, as shown in Table 9. Various components of verification such as semantic, syntactic, pragmatic and empirical verification are introduced in Section 4.7 of Chapter 4. The semantic and syntactic verification of ontologies is done using reasoning engines. An empirical and syntactic verification of the model view specification is performed using EXPRESS parsers. Pragmatic validation is performed by making use of a model view developer plugin that allows users to work with SEM libraries for specifying model views. Plans for long term test and implementation in the industry are also defined.

8.2 Verification of Ontology Definitions

8.2.1 Checking for Inconsistencies using Reasoning Engines

Formally specified ontologies can be verified using reasoning engines. Protege-OWL provides this facility through RACER [59].The objective is to verify the OWL syntax as well as the implicit semantic meaning associated with IFC entities included in the ontology being developed. These are defined in the form of DL syntax and associated with classes and relationships as explained in Chapter 7. For example, for a building element we can specify

Table 9: Proposed evaluation metrics and criteria description for a *SEM* based model view development framework.

Criteria	**Description**
Expressiveness and Rigor	By using a formal SEM structure, semantics in MVD aspects can be represented in a consistent manner.
Understanding complex views	Model views represent different levels of detail. SEM based development methodology contributes to a better understanding of model views by providing a concise, object oriented view of the exchange. A view is decomposed in several smaller modular objects that are manageable.
Traceability	Traceability is a very important feature in the development process. A more effective translation of the user needs into the design of MVDs that allow transparency is needed.
Quality	Better quality of MVD design may be achieved by using SEMs that are tested and verified.
Development time and costs	Unnecessary iterations and redundancy avoided due to the front-loading of concept design. Costs are also reduced by early verification of SEMs,
Reuse of SEMs	Building a new MVD becomes a matter of combining and configuring predefined components from a SEM library.

the following:

- There should be a placement relationship and representation associated with it. Also,
- It is disjoint with entities like transport element, furnishing element, distribution element, other element components etc.

The research plan was to check the consistency of classes defined based on the restrictions provided. Protege has the ability to link to external reasoning engines. The methodology involves defining rules in a formal and reusable manner. The reasoning engine is invoked to check the relations and the necessary and sufficient conditions. To demonstrate this approach a Probe Class is created, intentionally making it inconsistent to the definitions. For example, an instance of a piece is created as a subclass of Building Element. Later, while defining it the user classified it as a Discrete Accessory. Figure 58 shows the PieceInstance, which now has both *IfcBuildingElement* and *IfcDiscreteAccessory* as superclasses. When the reasoning function is performed it throws inconsistency and error is reported. Figure 58 depicts the hierarchy of entities when a probe class is inserted, and Figure 59 shows the inconsistency highlighted when the automatic reasoning is performed.

The reasoning methodology used in the above example is that *IfcBuildingElement* and *IfcDiscreteAccessory* are defined as disjoint classes; hence an new class instance created cannot be a subclass of both of them at the same time. Intuitively a piece cannot be a building element and a discrete accessory at the same time. However, in the case of large ontologies such issues become intractable and automated reasoning is the ideal solution.

Based on the semantics of classes and relationships and the necessary and sufficient conditions defined, automatic classification of entities is performed. It is possible to infer that, if Class B is a member of Class A, it has to satisfy condition 1, condition 2, condition 3, etc. However, a question arises. *Is it possible to say that if a class satisfies condition 1, condition 2, condition 3, etc it is a member of Class A?* This is an important step to define subsumption relations. Hence, the objective is to include semantics, which are not explicit within the IFC definitions also in the ontological definitions so that reasoning engines can achieve closure in terms of the Open World Reasoning ideology.

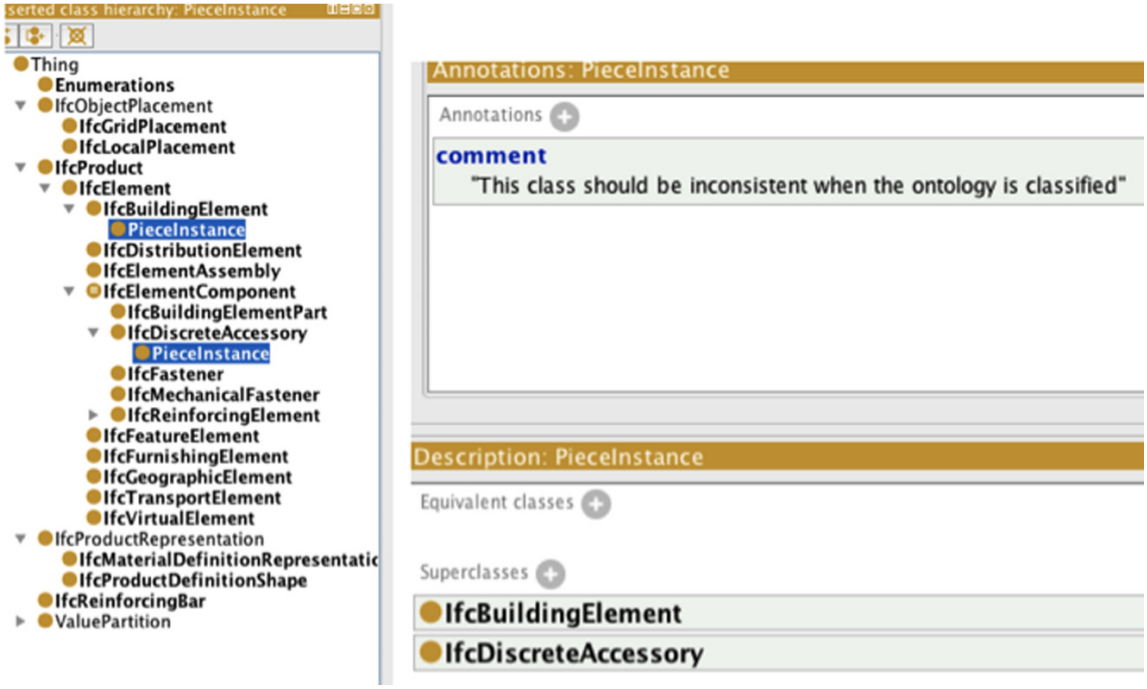

Figure 58: Hierarchy of entities with the probe class inserted.

Automated classification is not in the scope of the verification plans, but some attempts are made to try this. Figure 60 shows the early attempts at automatic classification. A new entity called *IfcReinforcingBar* is created and this is not inserted within the existing hierarchy. Instead it is created as a stand-alone entity and the semantics are defined. The following conditions are provided:

1. We know that *IfcReinforcingBar* has some representation, hence provide that as a restriction.

2. Also, we can make the assumption that the Geometry is Extrusion in this case. Include this as additional information and,

3. It is used as Reinforcing Element. Hence provide the condition that it is a typeOf.

The reasoning engine is invoked and run for consistency and classification. It returns no error and successfully inserts the *IfcReinforcingBar* within the defined hierarchy automatically. Figure 60 (a) show the hierarchy defined by the user called as asserted and Figure 60 (b) shows the hierarchy inferred by the reasoning engine. By setting up test cases for different

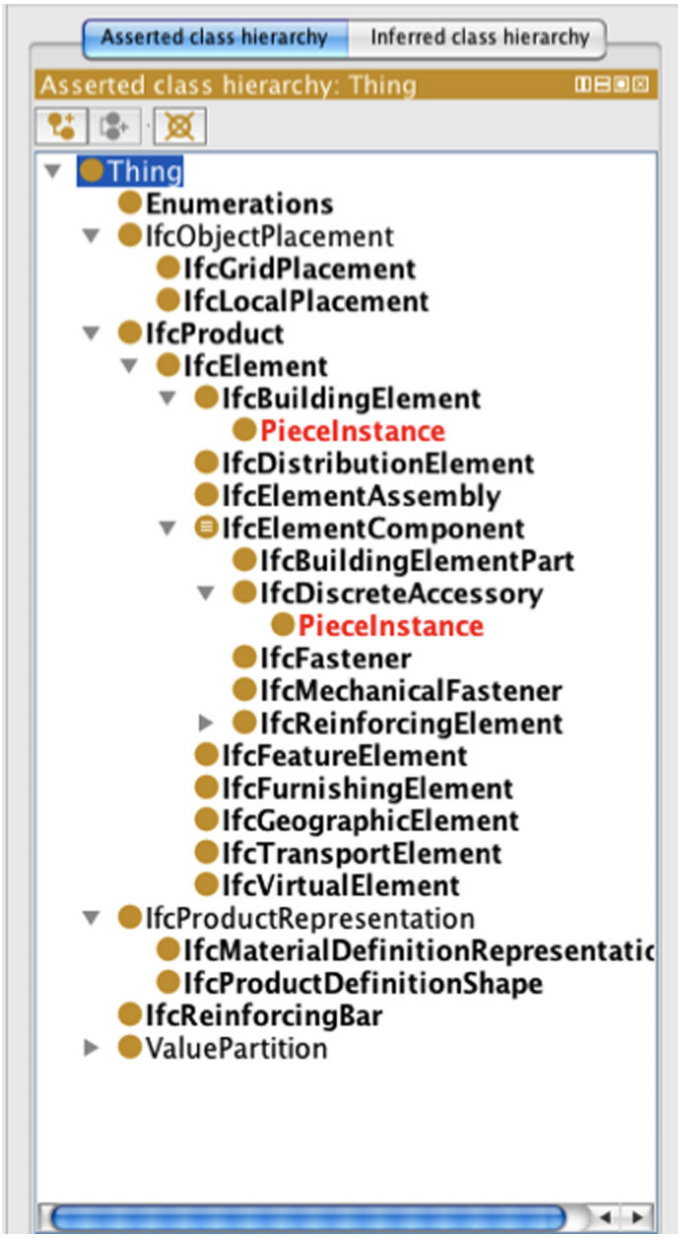

Figure 59: Hierarchy of entities with the inconsistencies highlighted after running reasoning engine.

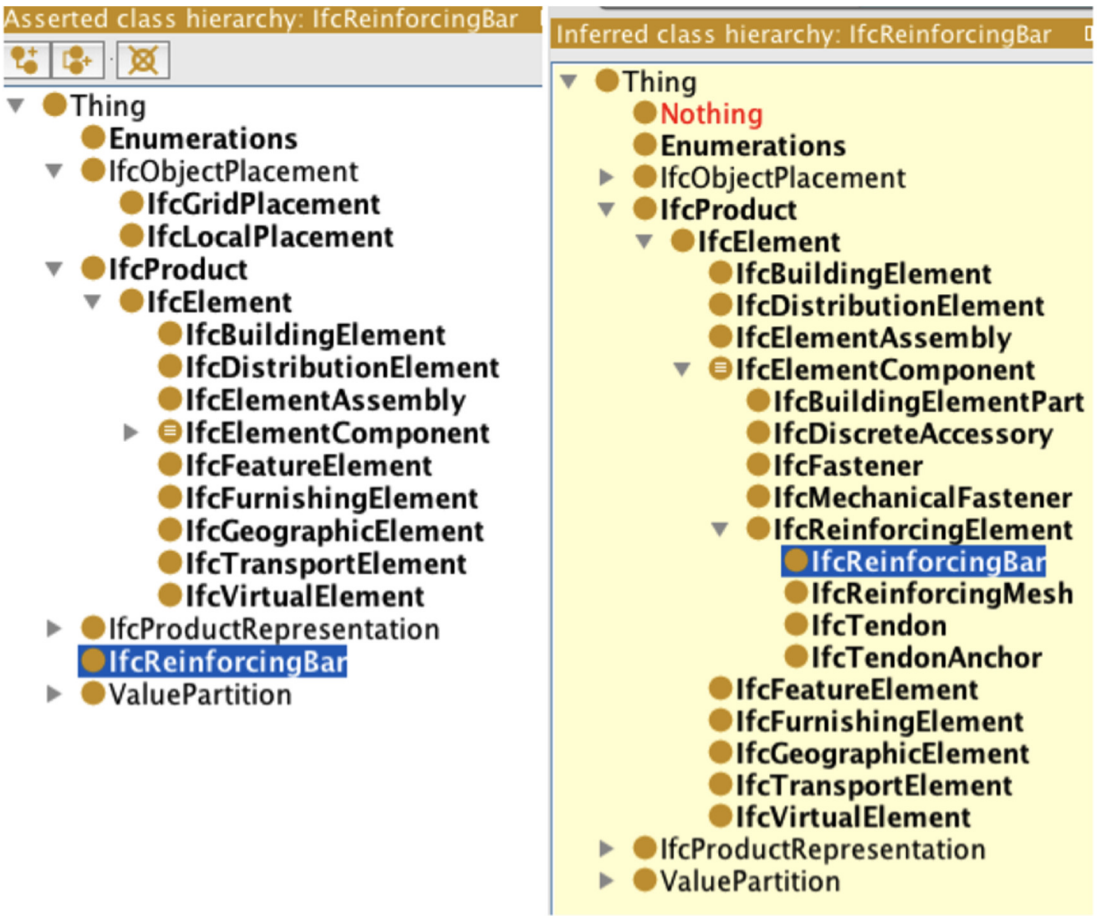

Figure 60: Automatic classification a) *asserted hierarchy* and b) *inferred hierarchy.*

elements in the ontology and running multiple reasoning engine iterations, the ontology is verified for syntax and semantics.

8.2.2 Visualization of Relationships and Entities as Directed Acyclic Graphs

In addition to textual reports, the use of graph visualization tools to explain the results, the concept packaging, the subsumption relations, etc is performed. This will help the end-user (IFC implementors, software vendors, researchers, or any industry user) to identify the semantics involved with modeling in IFC with limited misrepresentation and errors. Figure 61 shows such an example of *IfcReinforcingBar* hierarchy as a graph. This can be seen as an empirical verification process to visually check for correctness.

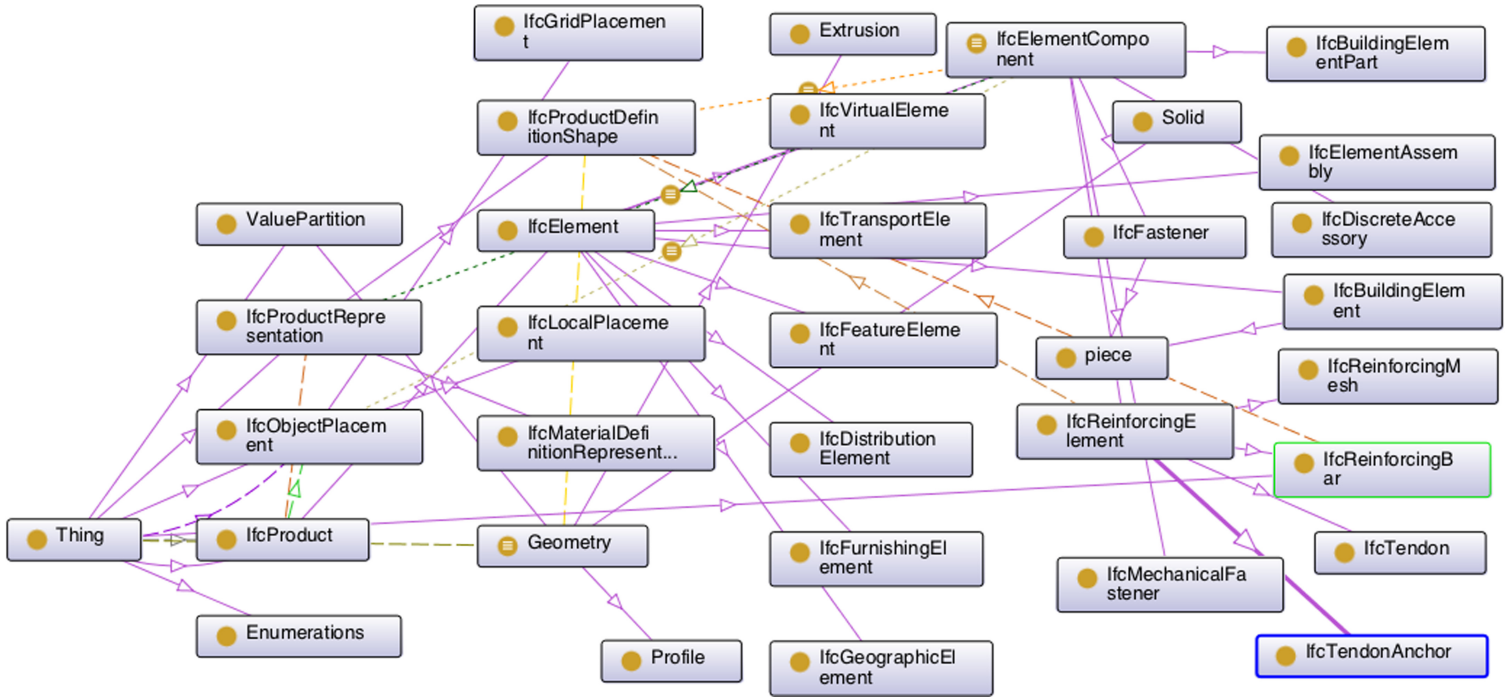

Figure 61: A graphical representation of IFC ontology with *IfcReinforcingBar* highlighted.

8.3 Test Cases and Validation Plans

This section shows how the ontology based approach and SEMs are useful for defining and improving the interoperability efforts. The objective of this effort is to provide a proof of concept by implementing a model view for a particular data exchange requirement in the AEC/FM industry. The model view is defined on the basis of SEMs, which are built on top of the engineering ontologies. The idea is to show the feasibility of modularizing the structure of model views into composable units. This necessitates precisely defining how precast concrete building model objects are to be exchanged using IFC files. For example,

- what combinations of IFC entities should be used to represent which precast objects?
- what the property values may be? and
- what IFC relationships are required between entities?

For this purpose, the ontology definitions in OWL are converted into libraries in C#. A model view developer plugin is written to make use of the libraries and provide users with the ability to specify modular MVDs. To fully illustrate the need for this research, a test scenario is shown in the following section, involving the export of a REVIT model into IFC schema and the subsequent IFC model into TEKLA Structures. The procurement stage of a precast project is used as the scenario and detailed studies are performed on precast double tees and other objects.

8.3.1 Workflow Description: MVD Authoring from SEMs

The process of generating a model view exchange is illustrated in Figure 62. The process begins by the user entering the needs for a model exchange in terms of SEMs. We assume that the SEMs already defined are sufficient at this stage to satisfy the requirements, which means the scope of the model view is restricted to the scope defined in this research. The user selects the required SEMs, for example, in the scenario shown in Figure 62, a precast double tee is to be exchanged with extruded geometry. The collection of SEMs selected has a mapping to the IFC schema, based on which an EXPRESS schema file is manually generated. EXPRESS syntax checkers are available as open source modules. EXPRESS

Engine developed by NIST is used in this research. The process of verification involves the following:

1. The EXPRESS schema file is parsed using the EXPRESS Engine
2. The errors are reported based on missing IFC entities, relations, and attributes
3. Modify the EXPRESS schema file for correct mapping
4. Repeat the steps 1 to 3 until all the errors are resolved.

This EXPRESS file, which is a valid subset of the overall IFC schema is used by the developers as the model view. This completes the syntactic verification of the model view generated from SEMs. A valid EXPRESS schema satisfying the Precast model view requirements is provided in Appendix D. At this stage, the model view generation is done by manually compiling the EXPRESS schema and checking it using parsers for completeness. This step of generating the schema based on user requirements can be automated by writing a schema generator. An automated model exchange protocol based on SEMs can be a future application as explained in the Section of Chapter 9.

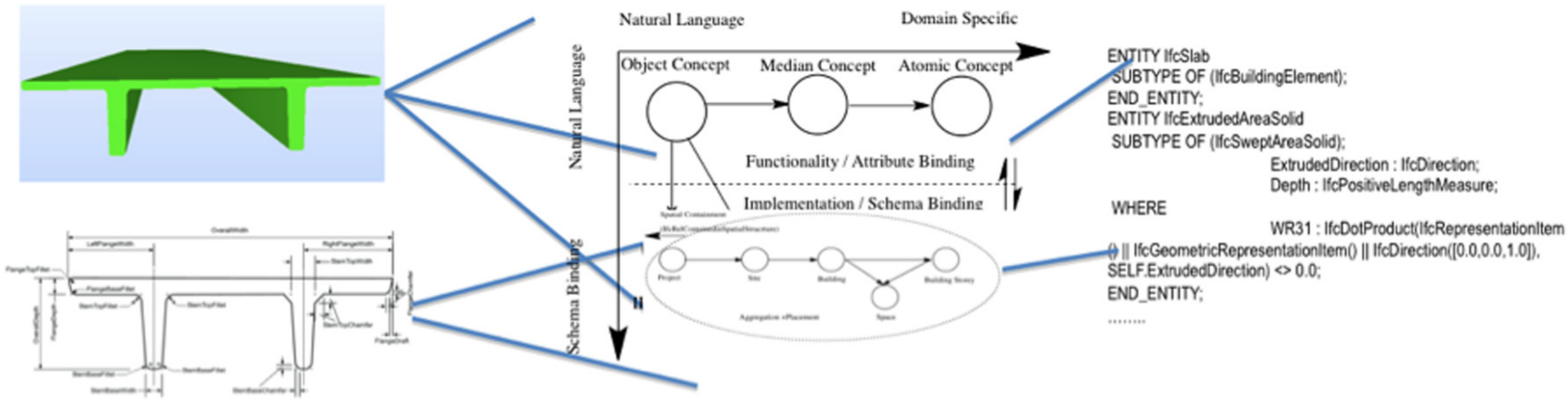

Figure 62: Steps to implement model views from Semantic Exchange Modules (SEM).

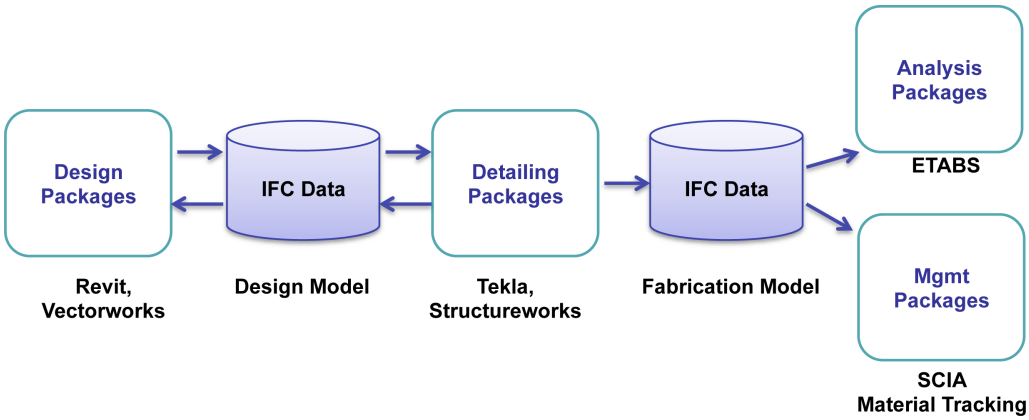

Figure 63: High level exchange scenarios investigated as part of this validation effort.

Appendix D shows the model view schema file compiled in EXPRESS format. This was tested for syntactic correctness using the EXPRESS Engine parser [95]. EXPRESS Engine is used to validate the model view file generated by running multiple iterations and this step completes the syntactic verification. Test cases are defined for validating the SEM based approach. Four particular scenarios involving Precast Fabrication Model, DoubleTee, Joint and Reinforcing Bar are explained in this section followed by the proposed methodology to be followed for IFC certification.

8.3.2 Test Scenario: Automation of Precast Procurement Process

The fabrication of precast pieces provides an opportunity for automation of the workflow in industry. For this to happen there is a need for seamless data exchange between the Architect and Engineer on one side to the Precast Detailer and Fabricator on the other side. The high-level model exchange scenarios are shown in Figure 63. The objective is to show the feasibility of modularizing the structure of model views into composable units. A test model is planned with a structure that is two stories in height. It is to be erected in the form of two slab bays; this makes four slabs total and 6 columns. Slabs on floor 1 are Hollowcore (HC) and slabs on 2nd floor are Doubletee (DT). The column spacing is fixed at 40 ft. by 50 ft. All beams and spandrels supported on columns by corbels. Both stories of slabs have a topping above DT and HC. Figure 64 shows the basic layout of the same in the form of a geometry only object created in AutoCAD.

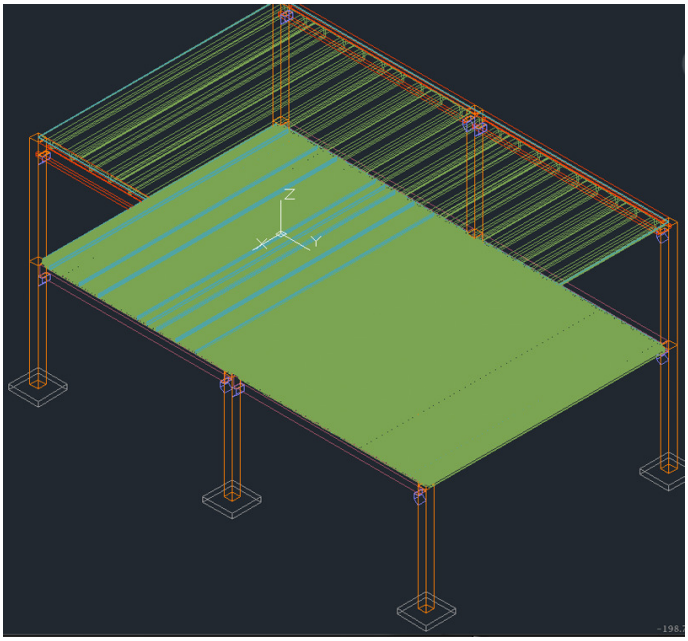

Figure 64: Test model created in AutoCAD for validation purposes.

8.3.2.1 Importance of Precast Fabrication Model

Traditionally, the exchange of data between architects and precast concrete fabricators occur in the format of *Contract Documents*. The architect provides the contract document to the general contractor, and passed on to the precast fabricator. This requires the precast fabricator to re-create all the information in the form of new set of drawings showing the details of precast pieces. This is called the *Precast Assembly Drawings* and is used for production of the pieces. The precast assembly drawings are passed back to the architect for design intent validation. The rework and time spent generating the same information in two different set of applications can be considerably reduced or eliminated completely by providing seamless interoperability between *Design Applications* such as Revit, ArchiCAD, Bentley and VectorWorks to *Detailing Packages* such as Tekla, StructureWorks, AllPlan etc.

The process flow diagram for a typical project where *Precaster* is a subcontractor is shown in Figure 65.

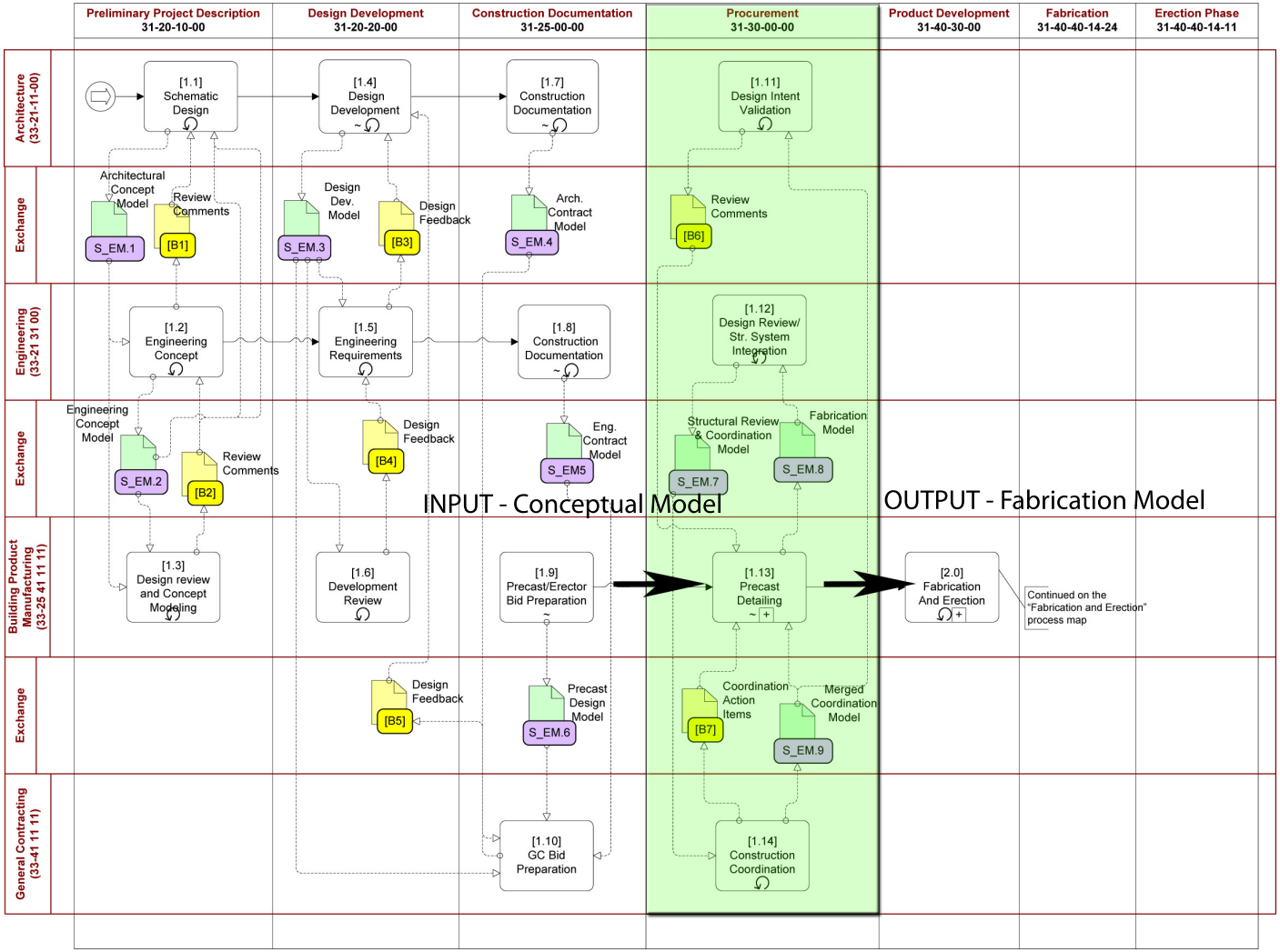

Figure 65: The work flows involved in a project, where Precaster is a subcontractor. The model exchanges involved in the procurement stage are highlighted.

Validation plans developed in this research focus on the procurement stage of the project *(Omniclass code: 31-30-00-00 Procurement)* that is highlighted in Figure 65. There are different tasks to be completed by various teams at this stage of the project. The following paragraph gives a brief description of the tasks and project teams involved.

The procurement stage takes the results of the *Precast Bid Preparation*, which falls under construction documentation. The Precaster prepares precast cost estimate based on the architectural and structural construction documents. The bid estimate may include schematic fabrication drawings, specifications and subcontractor procurement information. This will be followed by *Precast Detailing*, where a high level description of precast pieces is performed, including all embeds, reinforcing, tensioning cable layout, blockouts, finishes, joints, connections, lifting hooks for transportation etc. Structural engineers review the models and drawings prepared by the Precaster for structural integrity of the building. The *Design Intent Validation* will involve the review by architects or designers of the detailed fabrication model as required. It is the task of the general contractor to coordinate with all subcontractors regarding the sequence of construction and their delivery and erection sequences. Finally the fabrication and erection sequences are coordinated with all subcontractors. There is potential to automate all of the above steps such that model information is passed on between different stakeholders without loss of content and context.

Exchange models define the functional content of the information to be exchanged in a use case. For example, what will the geometry be used for editing, spatial coordination and clash checking, visual review etc.? Are internal embeds required or just geometry? In reality, there are varied exchange models such as Contract model, Fabrication model, Coordination model, Detailed Fabrication Model etc., involved in the procurement stage, as shown in Figure 65. But for the purposes of this test, these are merged into two main exchange models. The exchange model descriptions are described in the context of their project stage and exchange disciplines in a generic and high-level manner. The input data to this stage is assumed to be the *Conceptual Model* and the output data can be the *Fabrication Model*. Detailed requirements of these models are given in Tables 10 and 11. The

Table 10: Exchange model (EM) description for Precast Conceptual Model (Precaster as Subcontractor).

Project Stage	31-20-10-00 Preliminary Project Description
Exchange Disciplines	33-21-00-00 Architecture 33-21-31-14 Engineering 33-25-41-11-11 Building Product Manufacturing
Description	1. Purpose is to define/review basic structural system design, in terms of span directions, vertical transfers, major bracing, foundation pads 2. The conceptual model involves basic spatial layout of the building including external skin and building shape. It should identify potential locations for vertical load transfers 3. It might involve major architectural finishes, structural system selection, structural grid and site analysis. 4. Revit, Archicad, SDS/2, Tekla, STAAD Pro, RISA, ETABS 5. May be round trip or one-way
Related Exchange Models	A_EM.1, P_EM.1, S_EM.1

conceptual model is assumed to be the input rather than the contract model so that Precaster can provide the Architect feedback on the standard pieces types and profiles at the concept stage itself. This increases the collaboration in the project and reduces re-modeling at a later stage.

Model Preparation:

A *Conceptual Model* is created in REVIT that includes the basic geometry and spatial layout. This is shown in Figure 66. This model also shows that corbels are used as support. The model is exported into IFC (refer Table 12) and imported into TEKLA structures as a reference model. Different views of the model in TEKLA are shown in Figure 67. The piece detailing is performed in TEKLA, with standard profiles for HollowCore (HC) and DoubleTee (DT) pieces inserted into the model to replace generic shape. Shear (grout) keys

Table 11: Exchange model (EM) description for Precast Fabrication Model (Precaster as Subcontractor).

Project Stage	31-25-00-00 Procurement
Exchange Disciplines	33-21-00-00 Architecture 33-21-31-14 Engineering 33-25-41-11-11 Building Product Manufacturing
Description	1. The precast fabrication model is developed by the precastor and includes high-level description of precast piece detailing 2. The fabrication model involves all connection details, finishes, joints, embeds, reinforcing, tension cable layout, blockouts, pre-tensioned pieces, lifting hooks. 3. It is sent for review to the architect and engineer 4. Revit, Archicad, SDS/2, Tekla, STAAD Pro, RISA, ETABS 5. May be round trip or one-way
Related Exchange Models	A_EM.7, S_EM.8,

Figure 66: Conceptual model created with the basic structural system, spatial layout and building shape, rendered in REVIT.

Table 12: A subset of the Conceptual Model exported into IFC schema, showing the beams represented as BREP geometry.

```
#76635= IFCCARTESIANPOINT((4175.085,-816.72711,238.59382));
#76639= IFCCARTESIANPOINT((4175.085,-812.40688,257.57777));
#76643= IFCCARTESIANPOINT((4175.085,-801.26309,273.54243));
#76647= IFCCARTESIANPOINT((4175.085,-784.93258,284.14287));
#76651= IFCPOLYLOOP((#70483,#70498,#70528,#70558,#70441,
#70437,#76635,#76639, #76643,#76647,#70817,#76609));
#76655= IFCFACEOUTERBOUND(#76651,.T.);
#76658= IFCFACE((#76655));
#.......
#77034= IFCCLOSEDSHELL((#70338,#70361,#70376,#........ #77030));
#77038= IFCFACETEDBREP(#77034);
#77041= IFCSTYLEDITEM(#77038,(#63312),'Name');
#77045= IFCSHAPEREPRESENTATION(#29,'Body','Brep',(#77038));
#77051= IFCPRODUCTDEFINITIONSHAPE('','',(#77045));
#77055= IFCBEAM('19FIdF0005S34oCZWtD3Oq',#9,'021230','XTSIXSF143
3/4"*3 1/2"','XTSIXSF143 3/4"*3 1/2"',#70312,#77051,'TS_254590');
```

are included between the HC planks as shown in Figure 68. A topping layer is provided on DT and HC. Precast spandrel is resting on corbel connection attached to column, as shown in Figure 69. Liftinghooks are added to the pieces. The DTs, HC and columns are detailed with reinforcing elements. Edge conditions are checked so that there is no overlap of beam on column or topping on column. The entire depth of DTs are sitting inside the shelf and HC planks are cut to make provision for the column. These are some of the details included in the model to reflect a *fabrication model*. This is planned to be sent back to the A/E for design intent validation and review. The TEKLA model is exported into IFC and manually validated for conformance using a Solibri Model Checker. The IFC files had to be manually edited to match the required entities and relationships. Figure 70 shows the different details in the IFC model illustrated in Solibri Model Checker and Figure 71 shows the model imported back into REVIT to complete the model review process by A/E.

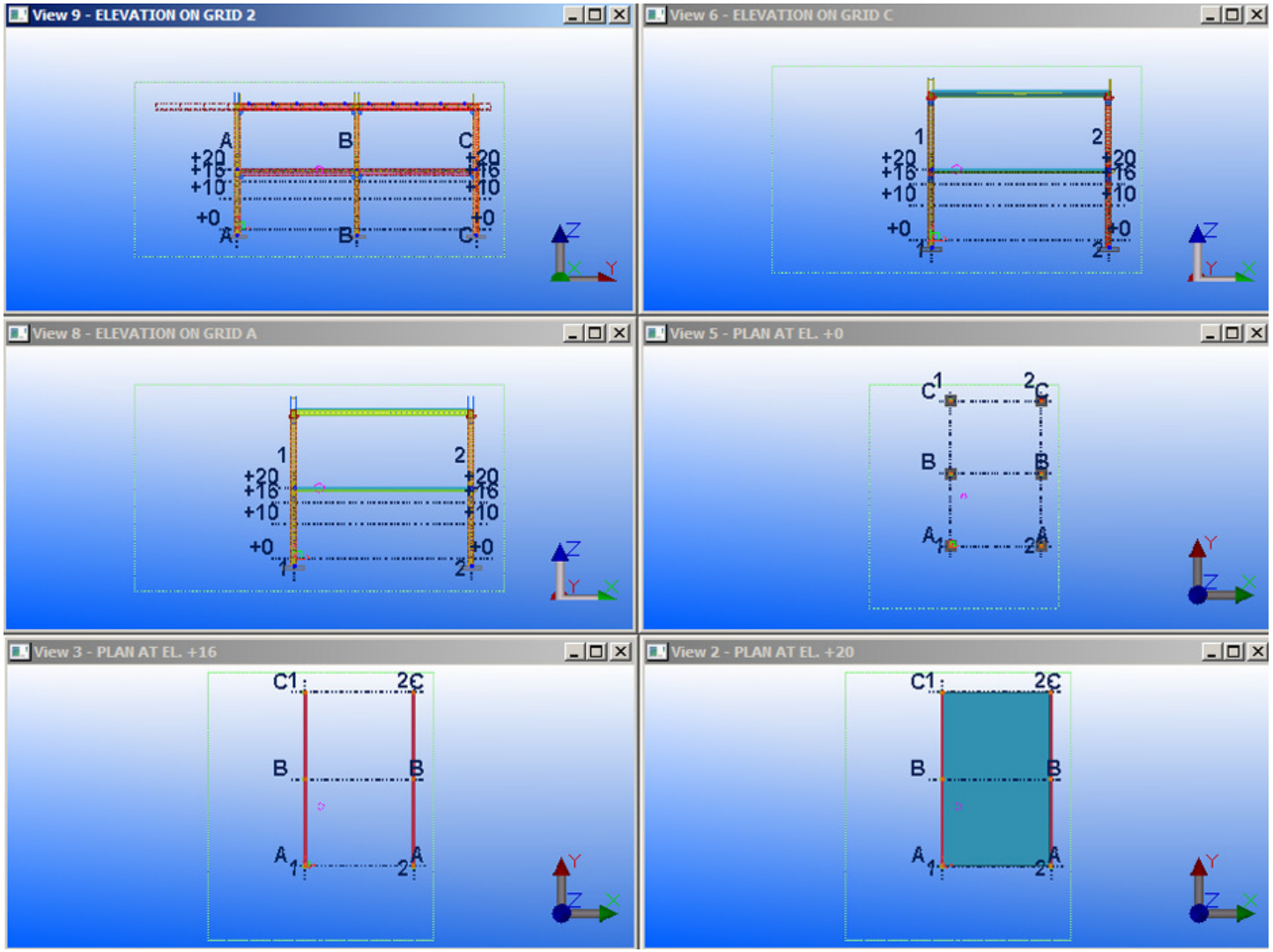

Figure 67: Test model created in TEKLA structures showing different views.

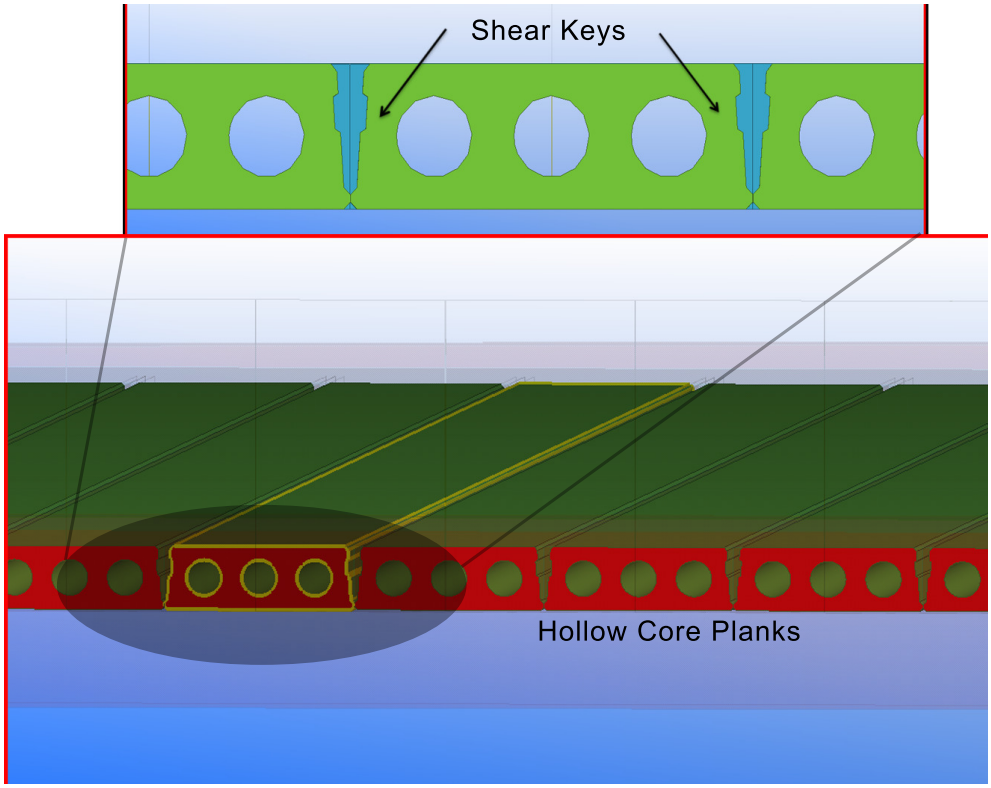

Figure 68: Detailed Model: Shear keys (Grout) inserted between HollowCore planks, done in TEKLA structures.

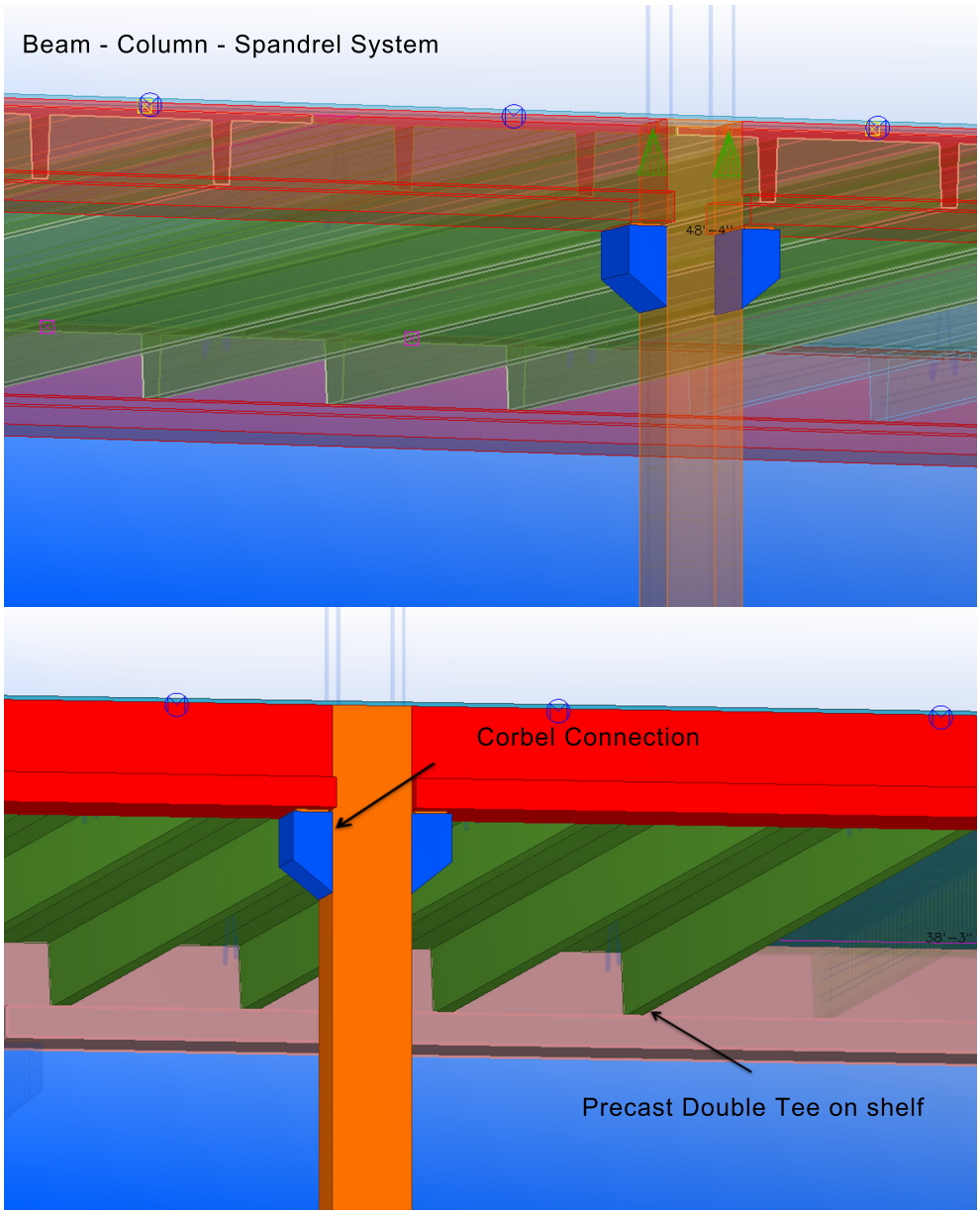

Figure 69: Precast beam-column-spandrel system with corbel connection.

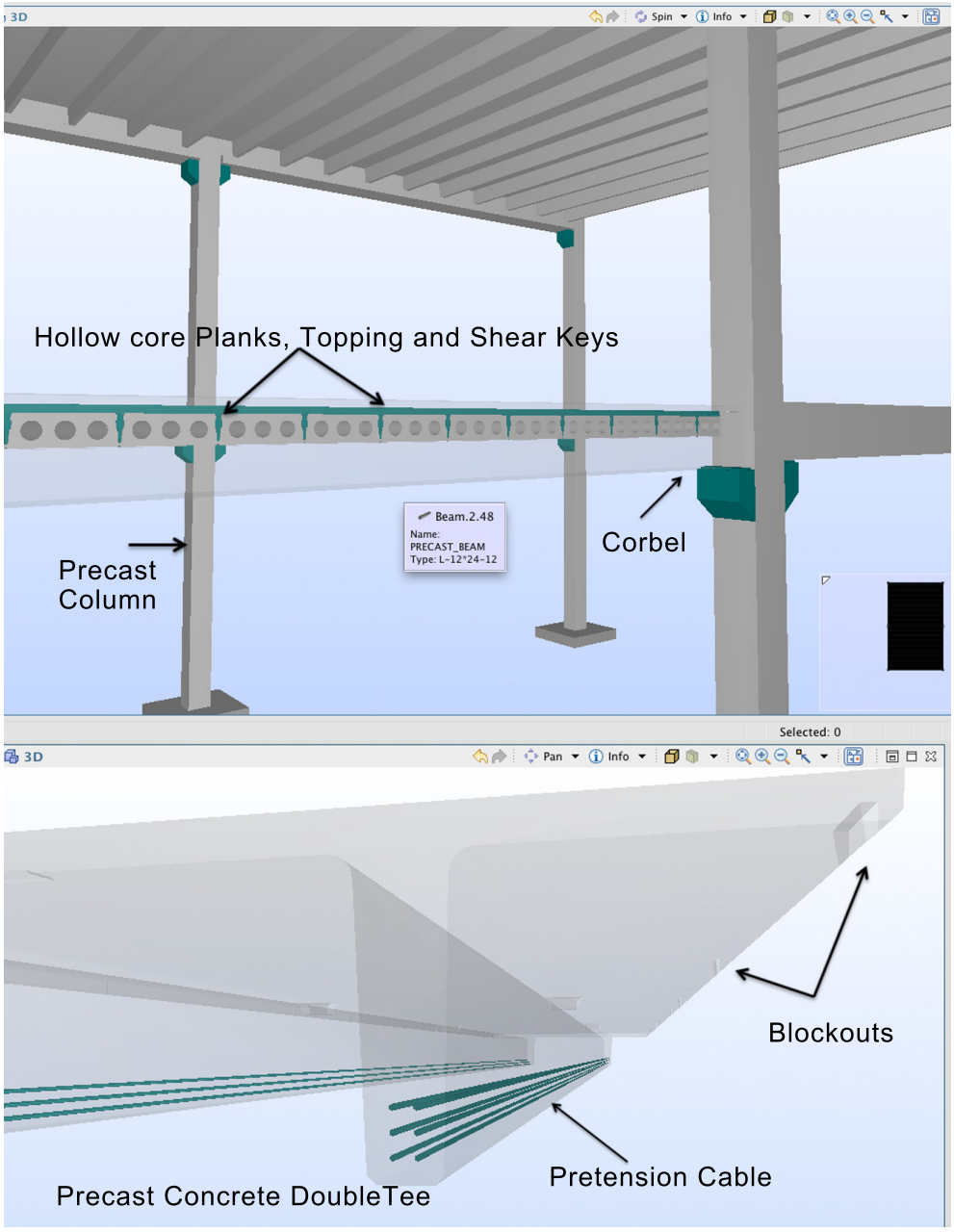

Figure 70: Fabrication model exported into IFC schema.

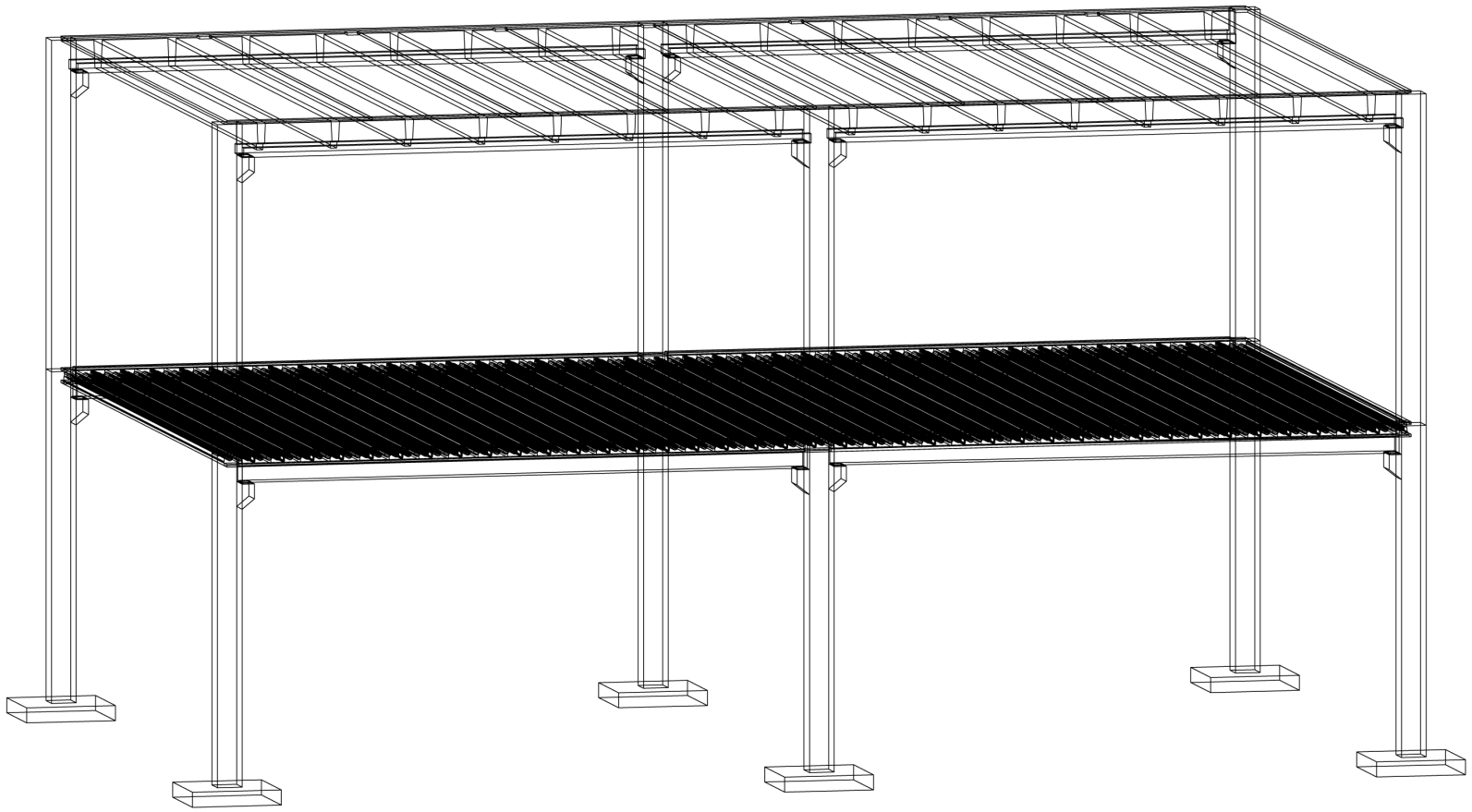

Figure 71: Fabrication model in IFC schema imported into REVIT.

8.3.3 Model View Developer Plugin

A plugin is envisioned as part of the BIM packages to implement the SEM based model view approach. This section illustrates the validation of the workflow using the model view plugin. C# is used to implement the plugin. C# was selected based on its capabilities, GUI representation and the fact that many BIM packages support C# based APIs. The main component to the model view developer plugin is the extendable SEM library written in C#. This is directly mapped based on the ontology definitions in Chapter six. This mapping is shown in Figure 72. The SEM structure is divided into three levels; the first level is comprised of high level objects such as building elements, discrete accessories, etc., the second level is made up of median SEMs that act like the connectors between the high level objects and the atomic entities that make up the third level.

8.3.3.1 Assumptions

The user is aware of the particular level of detail of model exchange required. SEMs are defined, implemented and available for use in export and import

- User deals with only the model objects and SEMs.
- The IFC entities, relations, and rules, which are based on ontology are hidden and works in the background.
- Model view definitions are generated in run time based on user input.

8.3.3.2 Process

Figure 74 shows the entry point of the model view developer plugin. The model exchange is usually performed by an expert who is aware of the exchange and business requirements. The user selects the objects in the native model and initiates the plugin. In Figure 73, a pair of precast DTs are selected by the user for exchange purposes. For example, if the exchange is made up of a beam-column system, then it is selected in the native model and then the model view plugin is run to create / specify the view. The GUI is made up of user input options in two columns. The left hand side consist of the options to select the domain

Figure 72: Hierarchy of the SEM structure implemented in C# library.

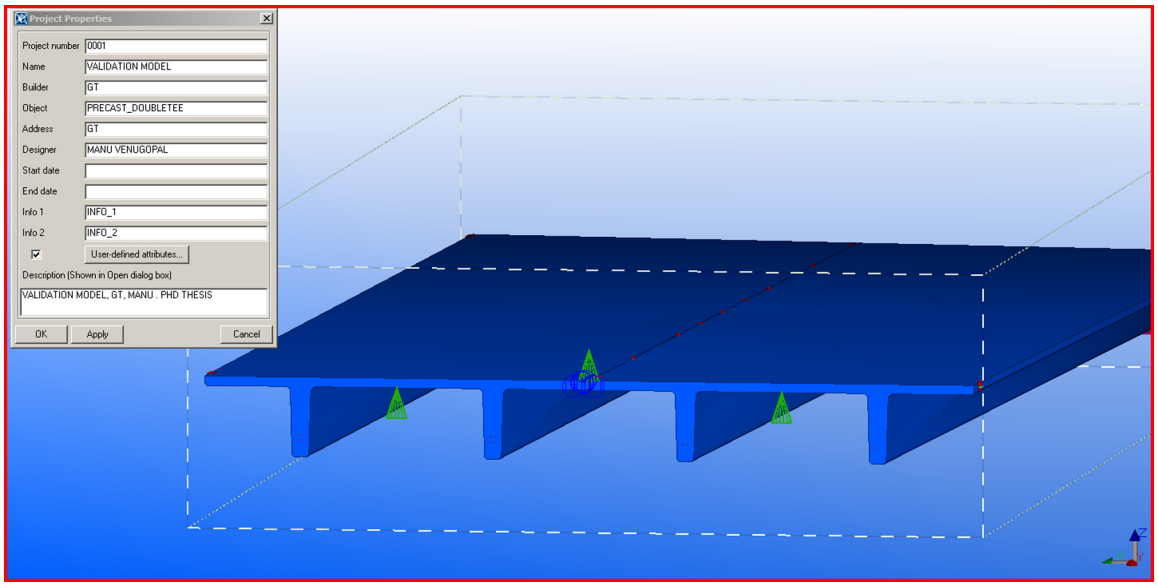

Figure 73: A precast double tee selected in native TEKLA model for model exchange validation using the model view plugin.

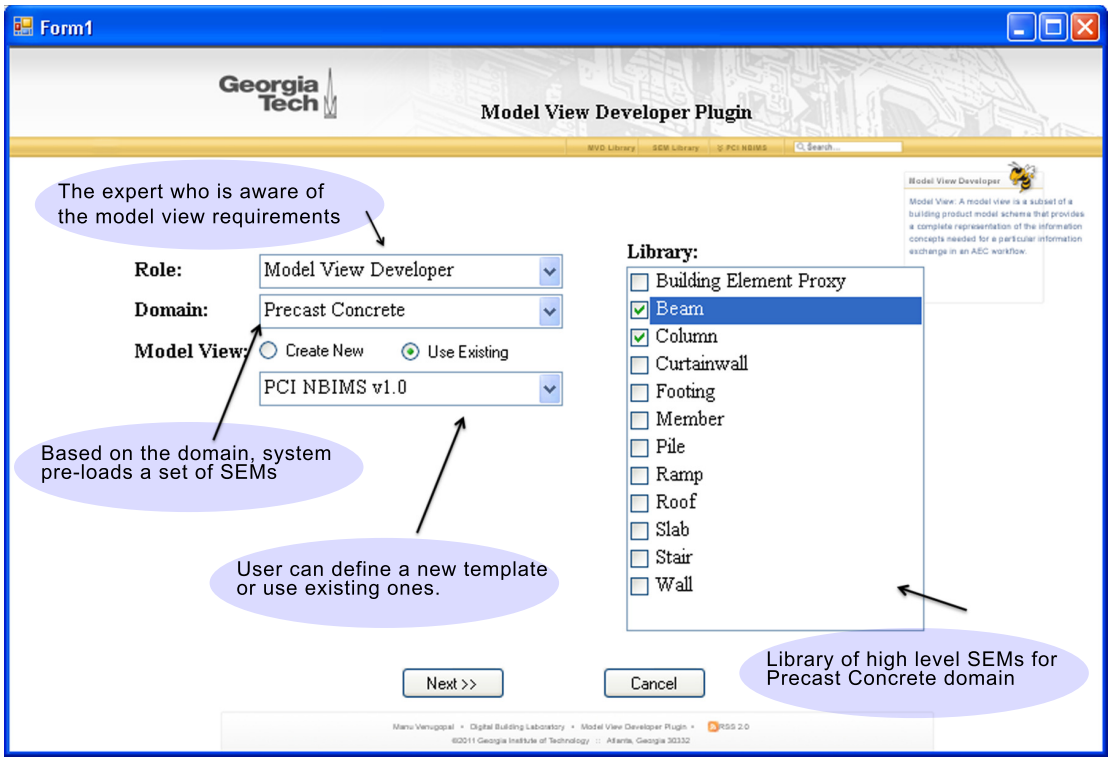

Figure 74: BIM model view developer plugin: entry point.

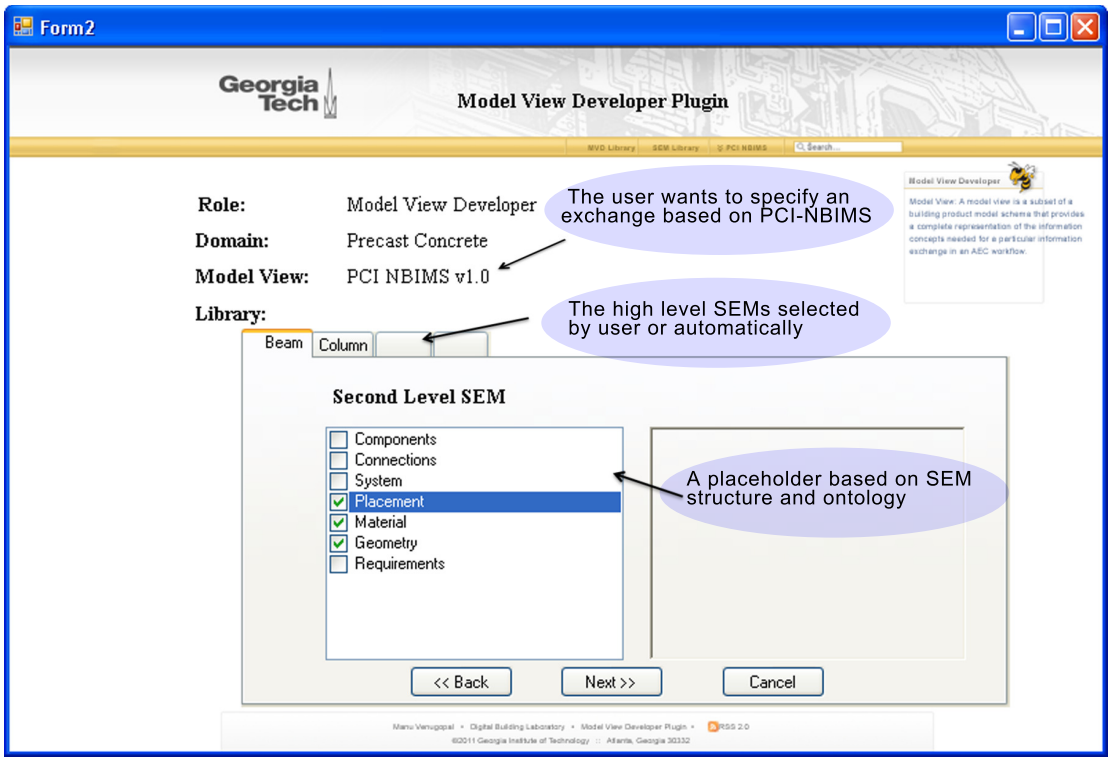

Figure 75: Second stage of BIM model view developer plugin.

and search for existing model views in the library. For example, if a user plans to create an exchange based on PCI NBIMS model view, then that can be selected from the predefined list. The right hand side consist of the high level SEM library, which is dynamically populated based on the domain selected by the user or automatically (in the future) based on objects selected in the model. The advantage of this approach is that the SEM structure is predefined and any model developed will be consistent with the usage of SEMs and thereby the mapping to the IFC entities and relationships will also be consistent. Moreover, the modular nature of the SEMs allow for reuse of same modules in different usage scenarios. For example, in the next step, the model view plugin asks the users to further define the attributes and information required in the model exchange. If a beam is selected as the high level object in the previous step, then the different menus available for beam are shown in the next step. Some of these attributes are required, whereas some are optional, as specified in the business rules. Different tabs are provided for each high level object as shown in Figure 75. If there are connection components (or secondary objects) to be attached, then

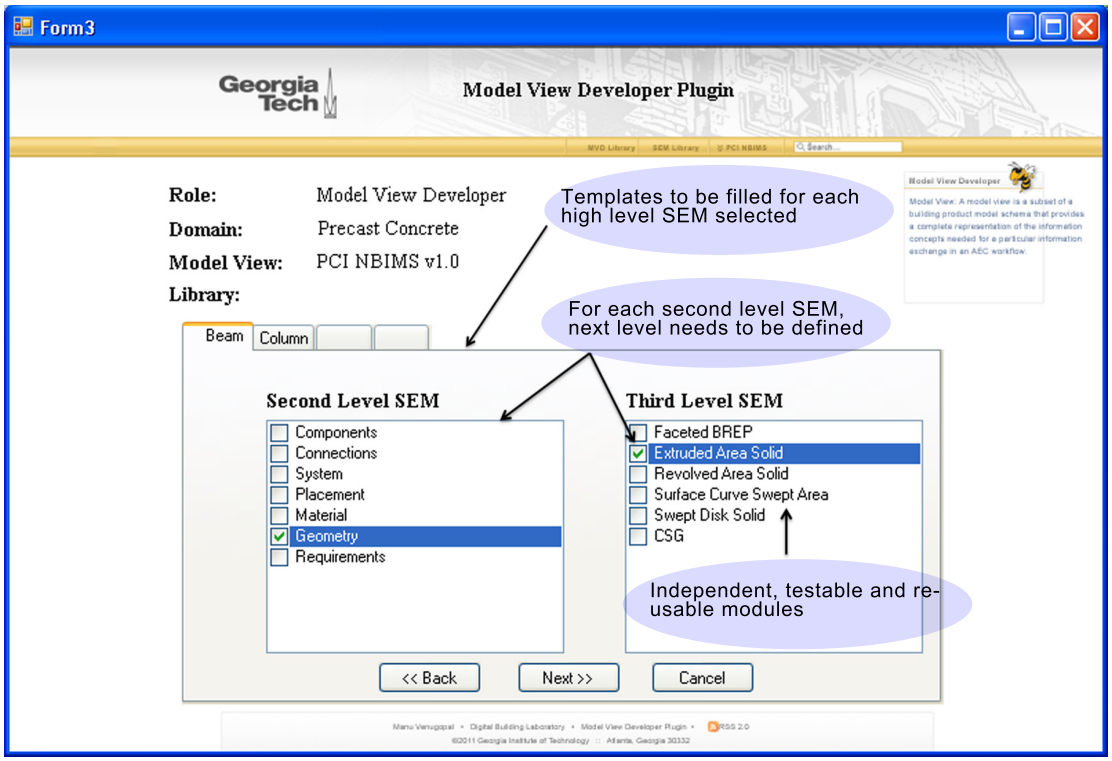

Figure 76: Selection of third level SEMs in MVD plugin.

an additional tab is automatically populated with the relationships to the parent element / assembly. For each of the second level SEMs selected for the exchange, system will bring up the third level list of SEMs required. These are the independent modules that have a binding to the IFC schema. For example, if Geometry is one of the second level SEMs selected then the system will ask, what type of representation is to be used; B-Rep, swept solid, CSG, etc. These same geometry SEMs can be reused for any other building element or discrete accessory as well. The following section provides a detailed overview of how this modularization is achieved for the Geometry SEM.

Geometric representation is assigned to any Product or Product Type. This SEM family is comprised of the assignment of representation to the Product, the type of representation, and how it is represented. Hence there are three levels of relationships to be checked in order to complete this SEM. For example, for any Product or Product Type, first it needs to be checked to see if there is a shape representation attached through a Product Definition Shape. Second, the Shape representation should point to the type of Geometry such as

B-Rep, Swept Solid, CSG etc. The last level should define how the product is represented in the form of surface, area, or faces etc. For example an *IfcProfleDef* defines the profile for a swept solid. The profile can be a parameterized profile, or an arbitrary profile etc.

The form of a building element is identified using the product definition shape entity. The geometry SEM has two main entities; *IfcProductDefinitionShape*, which provides a means to add a representation to a building element and *IfcShapeRepresentation*, which links to a specific kind of representation such as B-Rep, SweptSolid, CSG, etc. Multiple geometric representations of the same building element are possible by attaching a set of shape representations. Restrictions are to be applied such that there should be at least one representation and that only representation of type *IfcShapeModel (IfcShapeRepresentation or IfcTopologyRepresentation)* should be used to represent a product. The geometric representation types available in IFC are explained in Table 13, of which only the solid modeling types are considered in this regard.

One representation can be shared by many building elements by using an *IfcRepresentationMap* as one of the items through an *IfcMappedItem*. IFC provides several predefined types of representations of which the Solid Modeling types are B-Rep, Swept Solid, CSG etc., as shown in Table 13.

From the Geometry menu the user is asked to select the specific type of representation. For example, the DT beam is to be defined as extruded geometry. The swept area is defined by a cross section (also referred to as profile), which is given as a closed two-dimensional boundary on an implicit plane. A bounded planar surface is defined and inherited through the attribute *IfcProfileDef* (or subtypes) and this profile is extruded along a particular direction which is given by the attribute extruded direction (*IfcDirection*) for a given length / depth (*IfcPositiveLengthMeasure*). The swept area should be within the XY plane of the position coordinate system and the extruded direction should not be perpendicular to the local Z-axis.

A single high level object is fully defined before moving on to another object. Different tabs are provided for this functionality. Also, when related objects are present then the relationships are automatically set between tabs. Once all the entities, attributes, etc. are

Table 13: Geometric Representation Types available in IFC 2x3 data schema.

Type	**Description**
Curve2D	2 dimensional curves
GeometricSet	points, curves, surfaces (2 or 3 dimensional)
GeometricCurveSet	points, curves (2 or 3 dimensional)
Annotation2D	points, curves (2 or 3 dimensional), hatches and text (2 dimensional)
Surface Model	face based and shell based surface model
Solid Model	including swept solid, boolean results and B-Rep bodies
Specific types	
SweptSolid	swept area solids, by extrusion and revolution
B-Rep	faceted B-Rep's with and without voids
CSG	boolean results of operations between solid models, half spaces and boolean results
Clipping	boolean differences between swept area solids, half spaces and boolean results
AdvancedSweptSolid	swept area solids created by sweeping a profile along a directrix
Additional types	
BoundingBox	simplistic 3D representation by a bounding box
SectionedSpine	cross section based representation of a spine curve and planar cross sections. It can represent a surface or a solid and the interpolations of the between the cross sections is not defined
Mapped Representation	representation based on mapped item(s), referring to a representation map. It can be seen as an inserted block reference. The shape representation of the mapped item has a representation type declaring the type of its representation items.

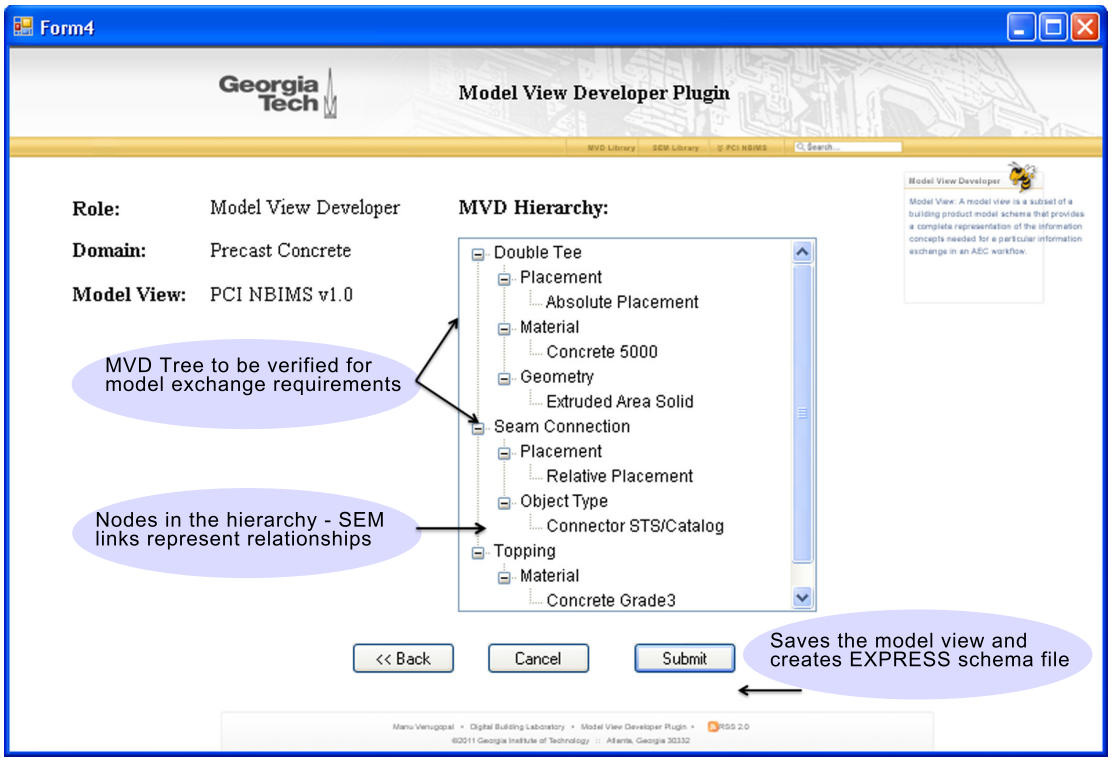

Figure 77: The tree structure of the model view specified in plugin.

fully defined the systems brings a summary of the model view specified in the form of a tree hierarchy, this can be used for a visual verification and also saved in the form of an XML schema. The final step in the plugin is to export the new model view generated in the form an EXPRESS schema. The exchange model is parsed into IFC schema using a schema generator conforming to the model view created or selected. Moreover, since the SEMs are assumed to be already implemented by both the exporting and importing application, the IFC file can be directly read into the importing application. Figure 78 shows the IFC file exported for the precast double tee from TEKLA imported into REVIT. This completes the model exchange scenario. At the moment the IFC files generated require manual editing as the software vendors are implementing the SEMs. Detailed implementation plans are given in the later section.

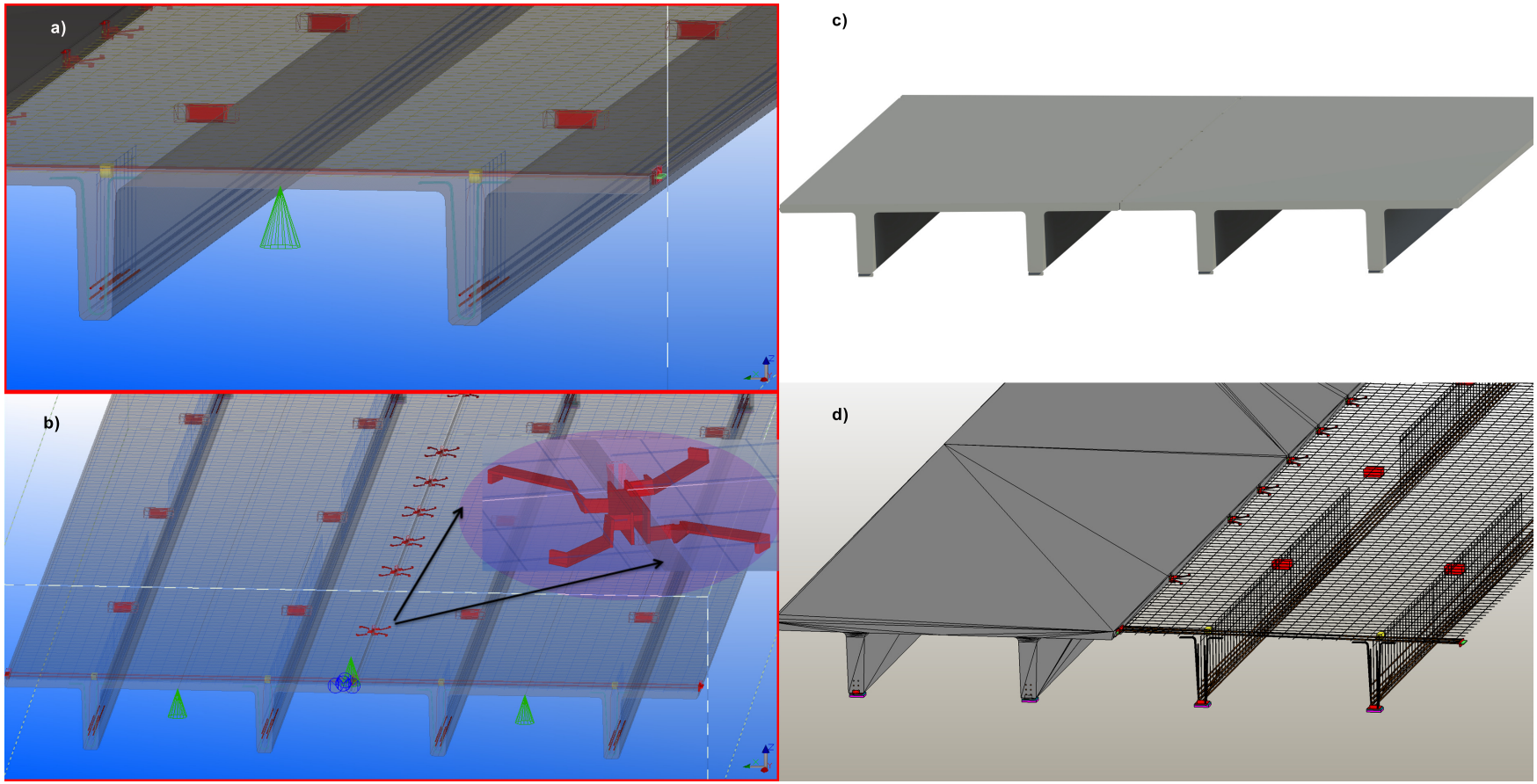

Figure 78: Precast detailed model in TEKLA (a & b) exported into IFC and imported into REVIT (c & d).

8.3.4 Validation of Model Progression and Level of Detail - Precast Slab Entity

Consider the case of a floor slab in a parking garage. A detailed precast design model is used for design intent validation by architect, for structural design review by the engineer, for coordination and clash detection by the general contractor and for production and fabrication sequencing by the plant manager. There are different ways to represent this slab entity using a product model schema such as IFC, depending upon the context and also the level of detail required. Figure 79 provides as illustration of the model with different levels of details at different stages of a project lifecycle. An IFC data model is at the center of all these exchanges and there needs to be semantic clarity about the exchange information content while the level of detail increases.

Five sample cases where the model exchange needs semantic clarity can be as listed below:

1. for purposes of clash detection among different disciplines such as MEP, or electrical, a simple boundary representation of the entire floor slab might be sufficient.

2. for structural analysis purposes the building components will have to be represented in the form of nodes and edges in a stick model (analytical model). There is no requirement for 3D geometry, however connections and loads (static and live) are important.

3. for precast fabrication purposes the slab would need to be represented in the form of individual hollow core planks with detailed geometry, relative layout, connection details and topping information.

4. a fourth case can be where there is a need for the parent slab as well as the individual hollow core planks, its topping, washes, and all components aggregated into the parent slab. In this case the geometry of the parent slab will be derived from the union of the individual components.

5. for production and delivery sequencing, there is no need for geometry information. However, the piece count and other information such as erection sequencing and project schedule are important.

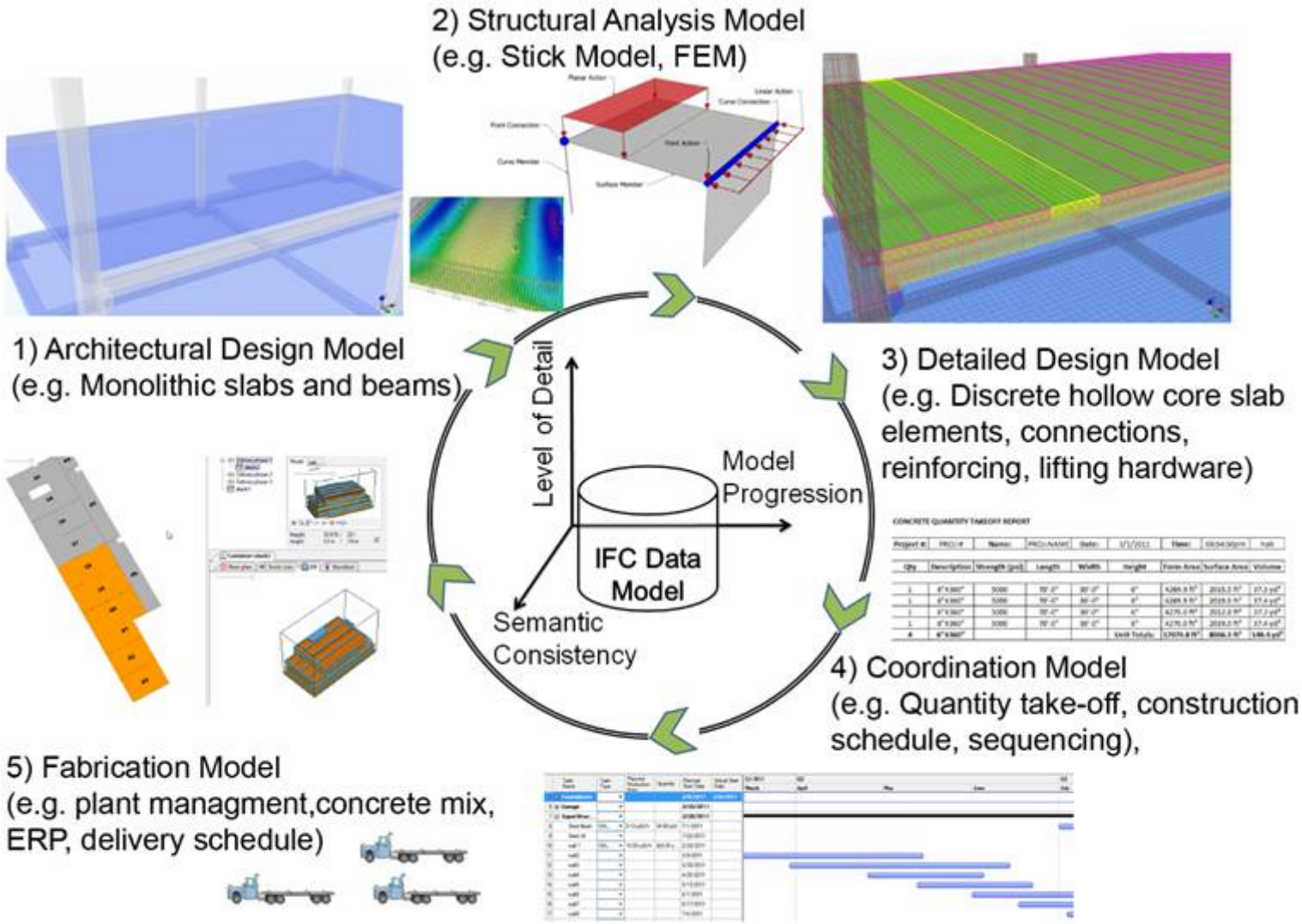

Figure 79: Issue of Model Progression and Semantic Clarity

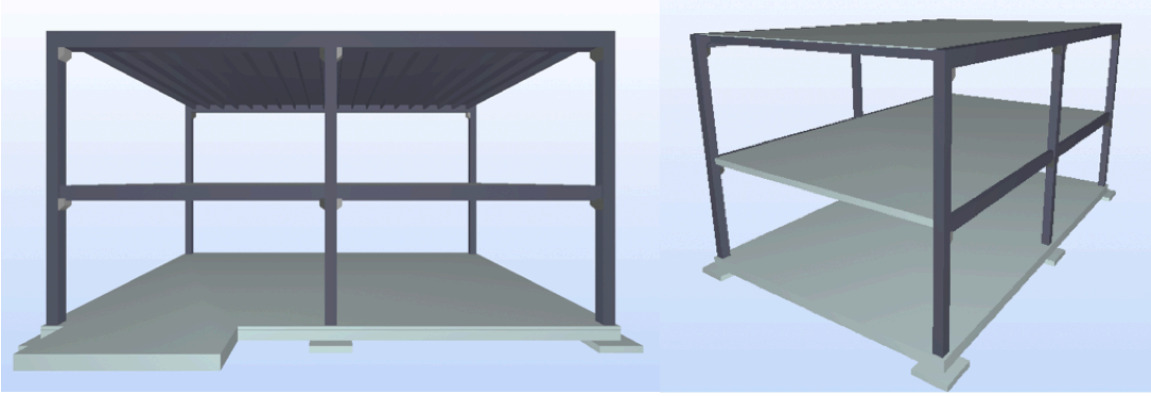

Figure 80: (a & b) Precast test model illustrating the monolithic floor slabs.

All five of the above cases can be represented in IFC and can coexist. This shows the richness of IFC as well as the redundancy due to the fact that it caters to a wide spectrum of the AEC/FM domain. Hence, effective exchanges require providing a layer of specificity over the top of an IFC (or any other) exchange schema. The purpose of the ontology structure and SEMs is to provide precisely such a layer. Therefore, it enables users to select and specify the appropriate information entities from a schema, their attributes, and rules that govern their possible values for particular uses based on SEMs without worrying about the underlying IFC schema. Each of the above four cases are explored in details below.

The use of a server with support for IFC schema is assumed in this study. Such a server can store multiple representations that are required in this case. This assumption is only to simplify the process, however all the five cases explained here can be dynamically derived each time from one single IFC file as well. By saving each of these queries or model views in a model server, they can be recalled with limited processing each time.

Case 1: Architectural Design Model: The architectural model created will generally have a monolithic slab entity that represents all the discrete elements that are its components downstream. There are two major use cases for exchanging this model. 1) to the structural engineer for structural design. 2) to precast detailer for detailed design of precast pieces. Figure 80 (a & b) shows the monolithic floor slab designed in the precast test model. Defining a model exchange using the MVD developer provides the SEM based menu structure for use. Let us assume the user selects the slab as a high level entity with B-Rep

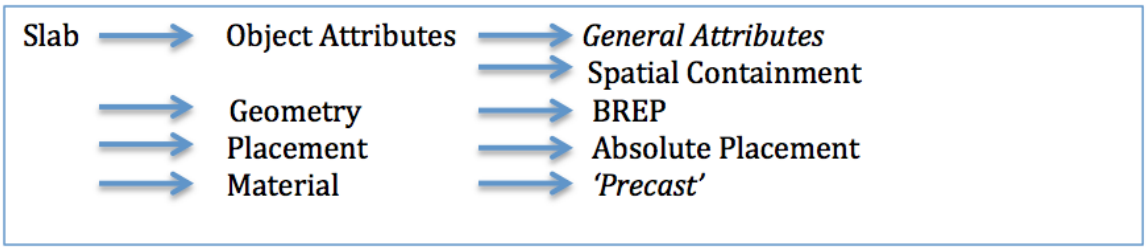

Figure 81: Menu structure for a Precast Slab to be exchanged from an architectural design model.

geometry and precast as material. The sample menu selection is shown below. It should be noted that Geometry, Placement, Material, and Spatial Containment are modular SEMs that are inherited by this Slab (building element) SEM. The system will generate the model view in EXPRESS schema for this particular case based on the mapping of SEMs to IFC. The IFC mapping is shown in Figure 82.

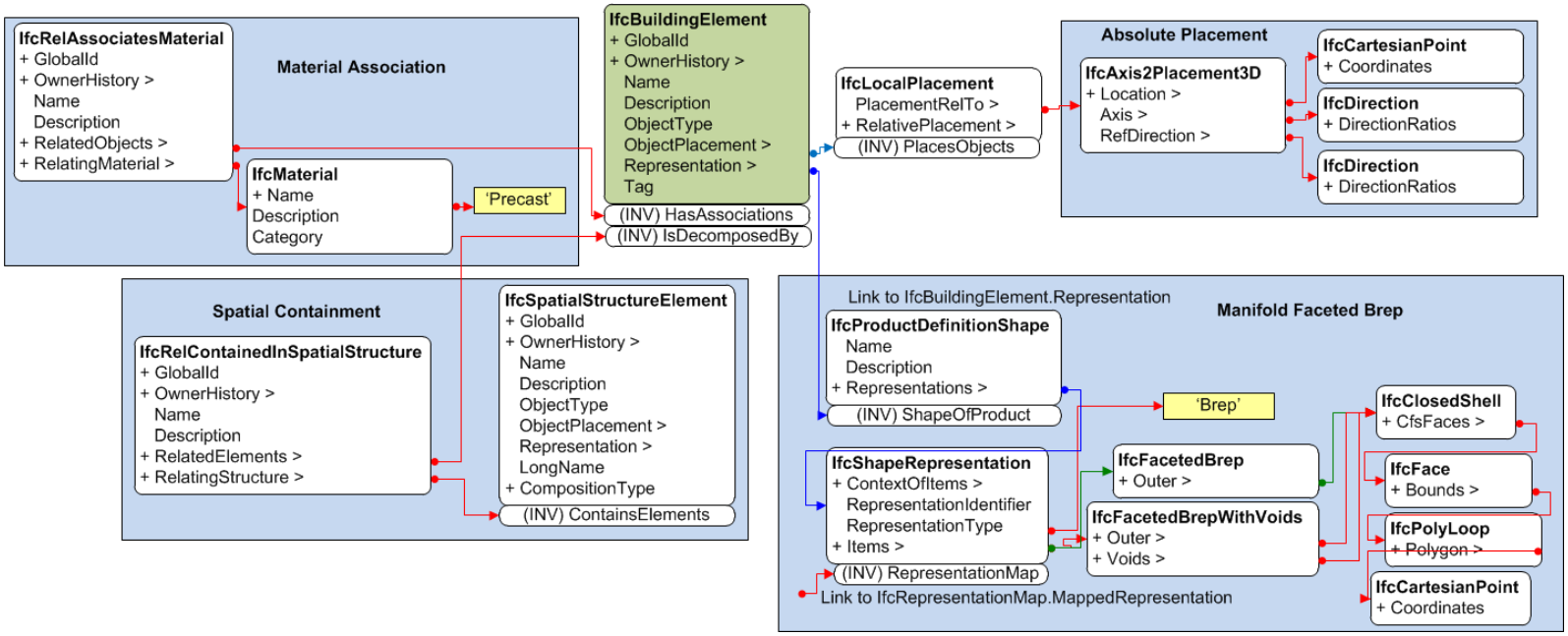

Figure 82: Ifc mapping for Architectural slab design.

The architectural design model had only a monolithic slab container for the second floor and generic double-tee slabs as placeholders for roof slab. Apart from the generic slabs, columns and beams, the architect has also designed corbels as features on column. It is important to note that these are designed as hierarchical objects such that corbels are a feature attached on the column. This architectural model is exported into Industry Foundation Classes (IFC) in two different representation formats, namely B-rep and Extruded geometry. B-rep and extruded geometry has different uses. B-rep is a simple face representation and is useful for volume calculations and clash detections. However, for more complicated tasks such as editing and parametric model progression there is a need for extruded geometry.

Case 2: Structural Analysis Model: The structural engineer designs the structural elements and also performs structural analysis based on the architectural model. An analysis model is usually comprised of nodes and edges. There needs to be a semantic matching between the analytical model and the physical model of a system. This can be achieved, based on the ontological definitions, using strong equivalency relationships. The slab geometry is converted into a linear member and mapped to a subtype of *IfcStructuralMember*. The structural member is embedded with an equivalency relation to the corresponding slab entity in the physical model, at the time of creation (of structural member) itself. Similar relationships are enforced for all the members in the model. This is equivalent to the sample shown in Case 1 of this section.

The engineer imports the entire Ifc model, however has control over which entities are to be imported and converted into native objects based on the SEM menu structure. (However, in the present form SEM library structure does not include loading conditions and reactions for members. These will be added in the future expansions.) Once this conversion operation is done, the engineer can perform various linear tests such as shear and bending and complicated analysis such as non-linear cracking, time dependent analysis, dynamics etc. Figure 83 a) and b) illustrates some of the analysis being performed on the precast piece members. In this demo SCIA engineer was used to perform the structural analysis tasks [37]. Based on the analysis results, structural piece members are modified for structural

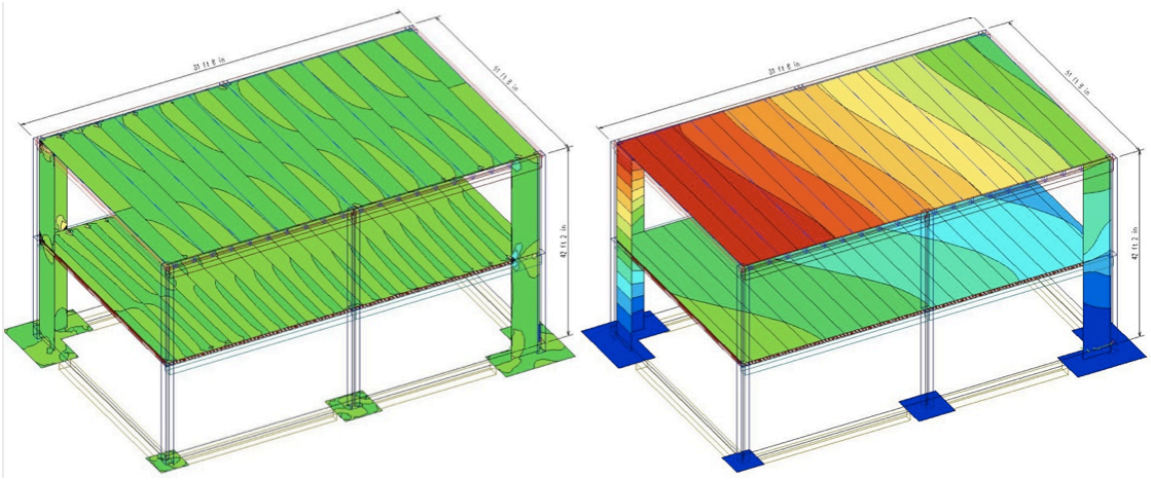

Figure 83: Structural analysis performed on the precast test model imported into SCIA engineer [37].

integrity. The structurally safe model, (after revisions) is passed back to the architect for review.

Case 3 and 4: Precast Detailed Model and Fabrication Model: The structural design model is passed on to the precast detailer. At the detailing stage accurate information about precast piece detailing as well as the connections, finishes, joints, reinforcing, tensioning cable layout, pre-tensioned pieces, embeds, lifting hardware, etc. are included in the model. In terms of model progression and level of detail, the detailed precast model should include all discrete elements when compared to the monolithic elements in design model. Hence, slab containers are replaced with individual precast planks, connections, topping and with provision for camber in-place. Figures 84 shows the details of individual Hollowcore planks in place of monolithic slabs as in Figure 80. The detailed model helps precaster to generate general arrangement drawings, assembly drawings, production drawings and the bill of material. There are two exchanges where the detailed model is passed back to the architect and structural engineer for design intent validation and structural review. Architects review the detailed model with corrections as required in terms of the joints and alignments of the precast panels, materials, topping and visible surface finishes. Structural engineers review the model for structural integrity. There are many cases in the industry where different precasters have their own standard member dimension. Hence,

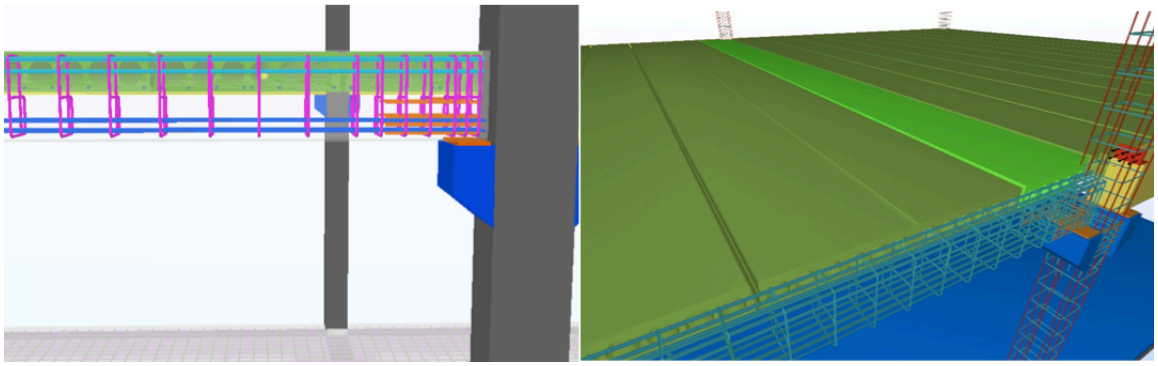

Figure 84: Individual hollow core planks in place of monolithic slab entity.

there might be changes in the thickness and width of the precast elements used by them. Therefore it is important for structural engineer to review the load bearing capacity of these modified pieces and also the temporary erection loads.

Let us assume the model view require the individual components such as the hollowcore planks, connections, and topping. The user can define this on the basis of the SEM structure. An example is shown in Figure 85. In a special case where the aggregated relationship is needed, only the individual hollowcore slabs are passed in the model exchange. In such cases, the parent slab entity will be missing and the model will consist of only the discrete hollowcores, connections, and toppings. The IFC mapping is as shown in Figure 86. There are different downstream uses for the detailed precast model. Construction coordination and clash detection, plant and part management, fabrication and erection, etc. are some of them. Advent of new technologies such as laser scans, point cloud-based models, and GPS, etc. are redefining the workflows in construction management. According to a McGraw hill survey (McGraw-Hill Construction 2009), the BIM-based coordination activities are contributing maximum benefit to the industry. In practice the general contractor brings together the models from different subcontractors and performs a spatial coordination between systems to avoid clashes before actual construction begins. B-rep geometry is sufficient to perform clash detection. Also, the contractor decides the construction sequencing and schedule. The final case is one such use of the detailed/fabrication model. **Case 5: Production and Delivery Sequencing:** The detailed precast model is also used by the plant management

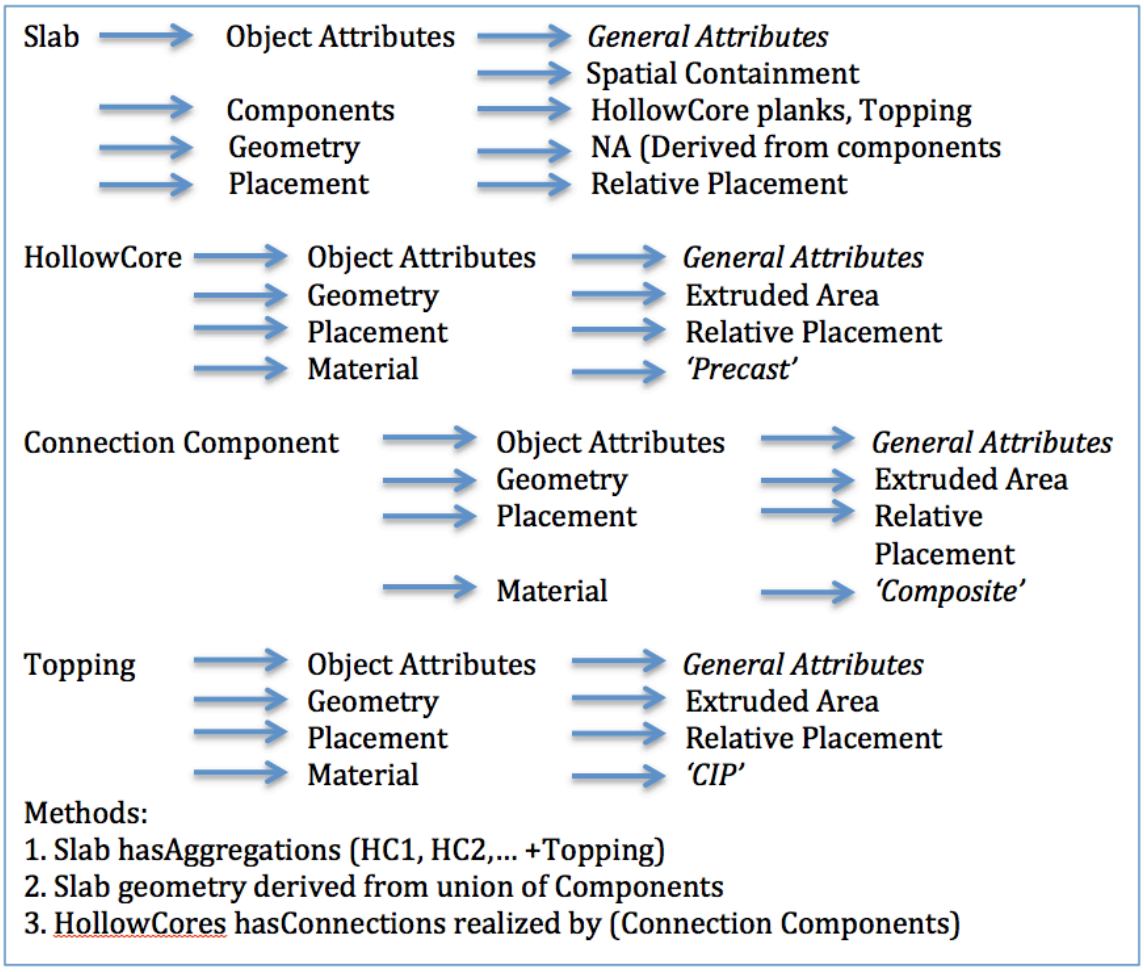

Figure 85: SEM menu structure for aggregation of slab components into parent slab.

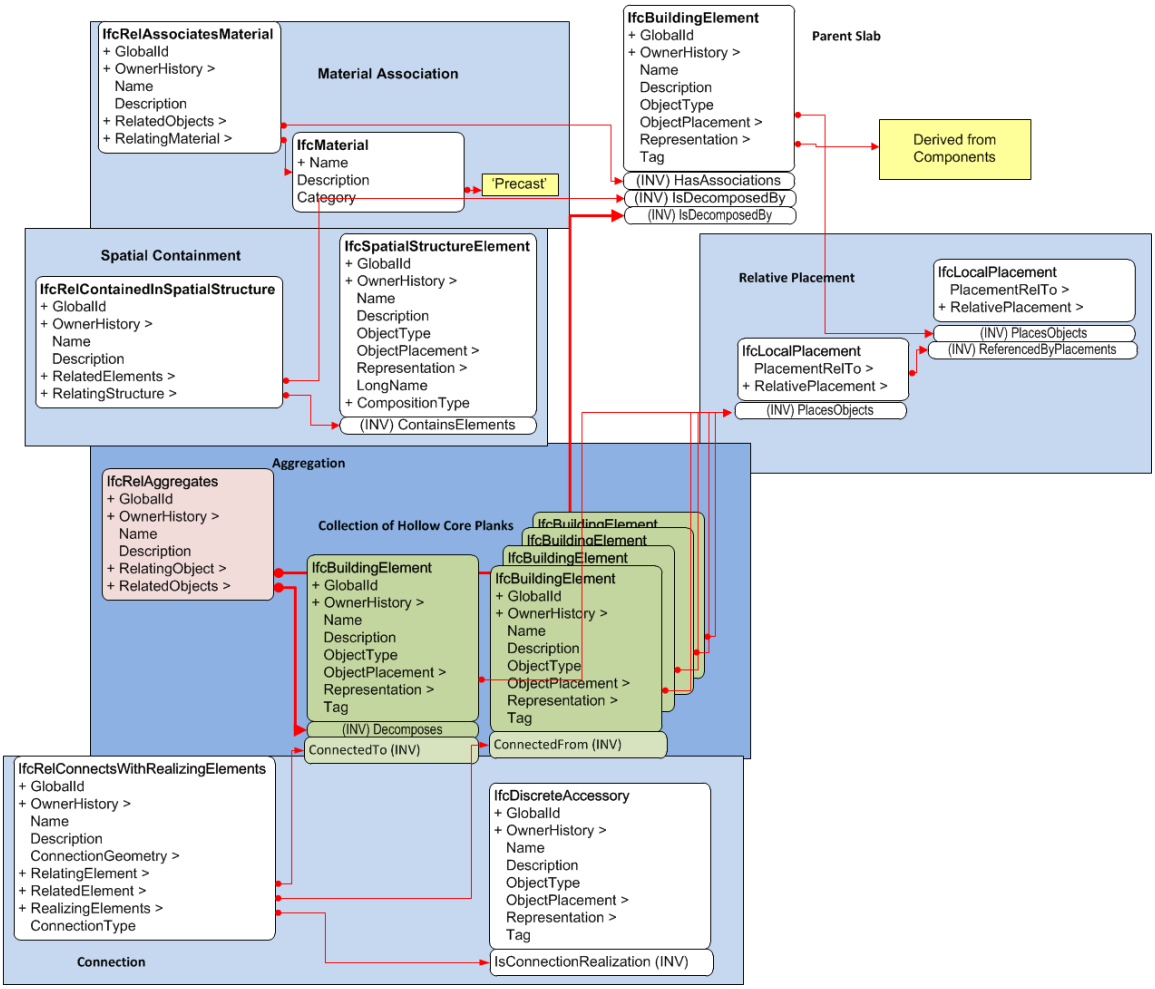

Figure 86: IFC mapping for aggregation of slab components into parent slab in precast detailed model.

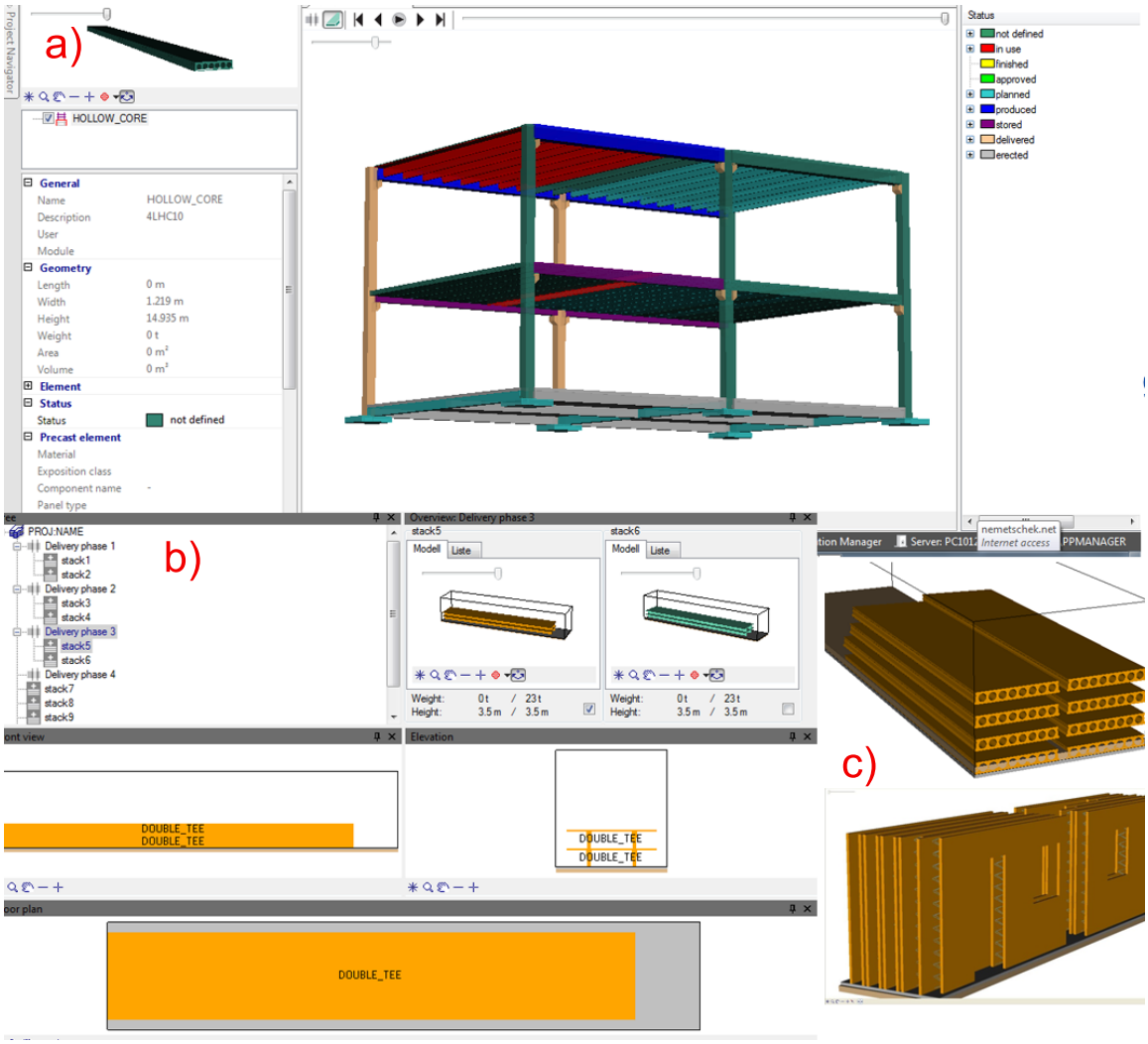

Figure 87: Simulating the a) production plan, b) delivery in precast part manager and c) stacking of precast pieces in the yard [37].

system to coordinate the fabrication and delivery of the precast pieces and also erection sequencing. These systems allow allocating parts to the fabrication beds as shown in Figure 87 based on the plant schedule and also orchestrate a delivery schedule. Testing was done on the SCIA Precast Part Manager system and Structureworks-Piecetracker [37]. Part management systems also integrate with ERP systems and can also pass information back to the general contractor. Production planning requires the delivery schedule of all necessary components and detailed product information including the layout, shape, material types,

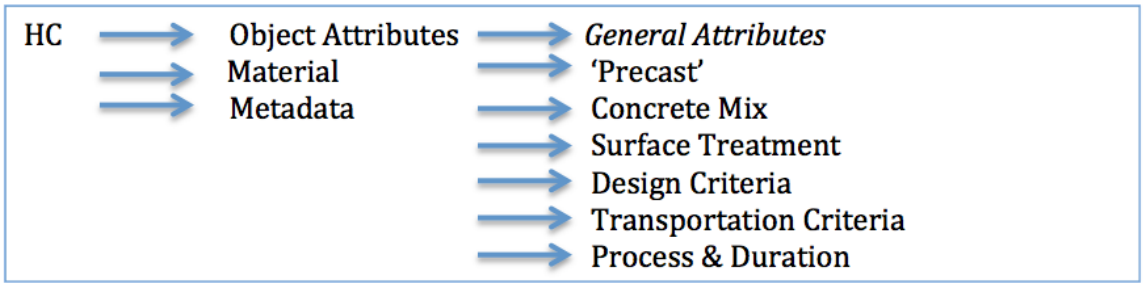

Figure 88: Menu structure for a Hollow core entity to be exchanged to a production planning system.

identification and product information, reinforcement specifications, assemblies and connections, concrete mixes and finish types, and lifting hardware. Therefore the information about hollow core planks are passed on in a flat format without geometric representation. Dimensions, quantity information, concrete mix, etc. are the main attributes. Further details about the test implementation of interoperability standards for the precast/prestressed concrete industry can be found in [37].

8.3.5 Test Case for validation of a Precast Piece Joint in Model Exchange

The required SEMS and rules for validating a precast joint in a model exchange is tested for correctness. A minimum subset of IFC entities required to satisfy the precast joint and the corresponding relationships are also provided. The business rules to be satisfied as part of validating the IFC entities for precast specific needs are provided as 8 major conditions. Each of these has a set of sub-rules to be satisfied. It is assumed that each of the entities and relationships listed here are completed without any missing references or pointers (Usually performed as part of any IFC syntax checking tool).

1. List of SEMs
   (a) Component
   (b) Connection
   (c) Placement
   (d) Geometry

(e) Material

(f) Type

(g) Requirement (Property Sets)

2. List of IFC entities required for the model exchange

   (a) Entities

      i. Valid subtypes of IfcBuildingElement

      ii. IfcFastener

      iii. IfcFastenerType

      iv. IfcElementQuantity

      v. IfcMaterial

      vi. IfcConnectionCurveGeometry

      vii. IfcLine

      viii. IfcRepresentationMap (depending on IfcFastenerType, see rules.)

   (b) Relationships

      i. IfcRelDefinesByType

      ii. IfcRelConnectsWithRealizingElements

      iii. IfcRelAssociatesMaterial

      iv. IfcRelDefinesByProperties

3. Rules

   (a) Condition 1: At least one instance of *IfcFastener* should be present that satisfies the following conditions for attribute values. There should be a total of 8 attributes, of which, the first 2 should compulsorily point to GUID and Owner History. Third and fourth attributes are optional and fifth attribute should be Object Type and contain string Precast Joint. There should be a reference linked to *IfcFastenerType* using *IfcRelDefinesByType*. Moreover, in *IfcRelConnectsWithRealizingElements* the ConnectingType attribute should match the Object Type

string, which is Precast Joint. The sixth attribute is Object Placement and should point to a valid placement entity. Similarly seventh attribute is representation and should point to a valid geometry concept. Eighth attribute is an optional Tag element.

(b) Condition 2: Should attach an *IfcMaterial* to the *IfcFastener* (Joint) through *IfcRelAssociatesMaterial*, for example, bituminous rubber for compression seal.

(c) Condition 3: Area, Volume, and Weight should be attached to the *IfcFastener* using *IfcElementQuantity* through *IfcRelDefinesByProperties*. Additional property sets (Optional) if existing should be attached through *IfcRelDefinesByProperties*

(d) Condition 4: *IfcFastener* should be assigned to subtypes of *IfcBuildingElement* through *IfcRelConnectsWithRealizingElements*. Some of the checks to be satisfied are explained as follows. There should be at least one instance *IfcRelConnectsWithRealizingElement* such that it connects the *IfcFastener* to two different Building Elements. The RelatingElement attributes should point to a physical piece which is a precast element and should not be a piece type. The RelatedElement attribute should point to any physical piece, precast or non-precast or members of other structural system. The connection geometry should define the line of joint and point to *IfcConnectionCurveGeometry*. The connecting type should be precast joint.

(e) Condition 5: There should be at least one instance of *IfcConnectionCurveGeometry*. The curve on relating element attribute should be used to limit the location and extent of the joint and should point to a valid *IfcLine* or *IfcPolyline* or *IfcCompositeCurve*. Curve on related element should be null as the curve is identical on both elements.

(f) Condition 6: Check for the existence of *IfcFastenerType*. If present it should satisfy the following rules. The *IfcFastenerType* should be linked to *IfcFastener* through *IfcRelDefinesByType*. GUID, OwnerHistory mandatory. Element type

should be provided. Possible values include; Vertical open-drained (Slotted neoprene baffle plus vertical air-seal), Horizontal open-drained (profiled with flashing and horizontal air-seal), Face Sealed, Compression Seal with gasket, Compression Seal with flexible material, or Other. The applicable occurrence can have four values (None, RelatingOnly, or RelatedOnly, or Both), based on which the representation maps attribute also changes.

(g) Condition 7: If FastenerType is present, then there should be corresponding instance of *IfcRelDefinesByType* relationship. The related objects and relating objects should point to instances of *IfcFasteners* and *IfcFastenerType* only respectively.

(h) Condition 8: Based on the value of Applicable Occurrences in FastenerType there should be 0, 1, or 2 instances of valid *IfcRepresentationMap* and they should point to a valid *IfcRepresentationMap* (as an initial test case). In reality this should map to a geometric set of 2D curves that define the cross-section profiling of the pieces on either side of the joint.

Figure 89 shows the visualization of the sample file developed for Precast Joint with an excerpt of the Part-21 specification. Another interesting test case that was developed comprised of a precast column with reinforcing bar in the form of extruded geometry. This is illustrated by Figure 32 (a-d) and explained in Chapter 5 (Section 5.4). Table 14 shows the Part-21 file sample developed for representing reinforcing bar as extrusions. The results showed the following:

1. Project-Site-Building-BuildingStorey hierarchy is successfully identified.
2. Grids are not supported in the export file.
3. Precast column is successfully represented in the form of extruded geometry.
4. Voids and openings are represented as generic objects with no semantics.
5. *IfcSweptDiskSolid* with *IfcPolyLine* as directrix and a disk radius works fine only for reinforcing bars with no bends.

Table 14: Part-21 file representation of Reinforcing bar in the form of extruded geometry using *IfcSweptDiskSolid* and *IfcSweptDiskSolidPolygonal* (IFC 2x4 entity).

```
#100350= IFCCARTESIANPOINT((15517.5,-330.,-330.));
#100360= IFCCARTESIANPOINT((15517.5,325.,-330.));
#100370= IFCCARTESIANPOINT((15517.5,325.,-30.));
#100380= IFCCARTESIANPOINT((15517.5,-330.,-30.));
#100390= IFCPOLYLINE((#100350,#100360,#100370,#100380,#100350));
#100400= IFCSWEPTDISKSOLID(#100390,6.,$,$,$);

/*replace above line with the following line,
 if ifcsweptdiskpolygonal is supported*/
/*#100400= IFCSWEPTDISKSOLIDPOLYGONAL(#100390,6.,$,$,$,10.);*/

#25024= IFCSHAPEREPRESENTATION
(#40,'Body','SweptDiskSolidPolygonal',(#100400));
#25030= IFCPRODUCTDEFINITIONSHAPE('','',(#25024));
#25034= IFCAXIS2PLACEMENT3D(#92,#465,#33);
#25037= IFCLOCALPLACEMENT(#79,#25034);
#25040= IFCREINFORCINGBAR('19w9$j0007QJ4oCpavEJ8u', #20,'Stirrup
L1-2','','',#25037,#25030,'TS_27053786',$,9.525,71.256,$,$,$);
```

6. *IfcSweptDiskSolidPolygonal*, with optional fillet radius will be better suited for rebar bending, but currently not supported. This can be tested only after IFC 2x4 is released and implemented.

7. Use of *IfcCompositeCureve* as the directrix with separate composite curve segments for the straight sections and the curved sections can be a work around for the problem.

The precast joint test case and sample files can be successfully visualized for verification. This test file is verified for syntax using the process explained in Section 8.4. However, validation is not possible for the test case with reinforcing bar as extruded geometry. This is owing to the fact that none of the available IFC viewers support reinforcing bar in the form of extruded geometry. Hence, the dependency on software vendors to implement specifications is identified as a major road block in the research progress. Test specifications similar to the ones defined above for Precast Joint, Column, Reinforcing Bar, etc., are developed

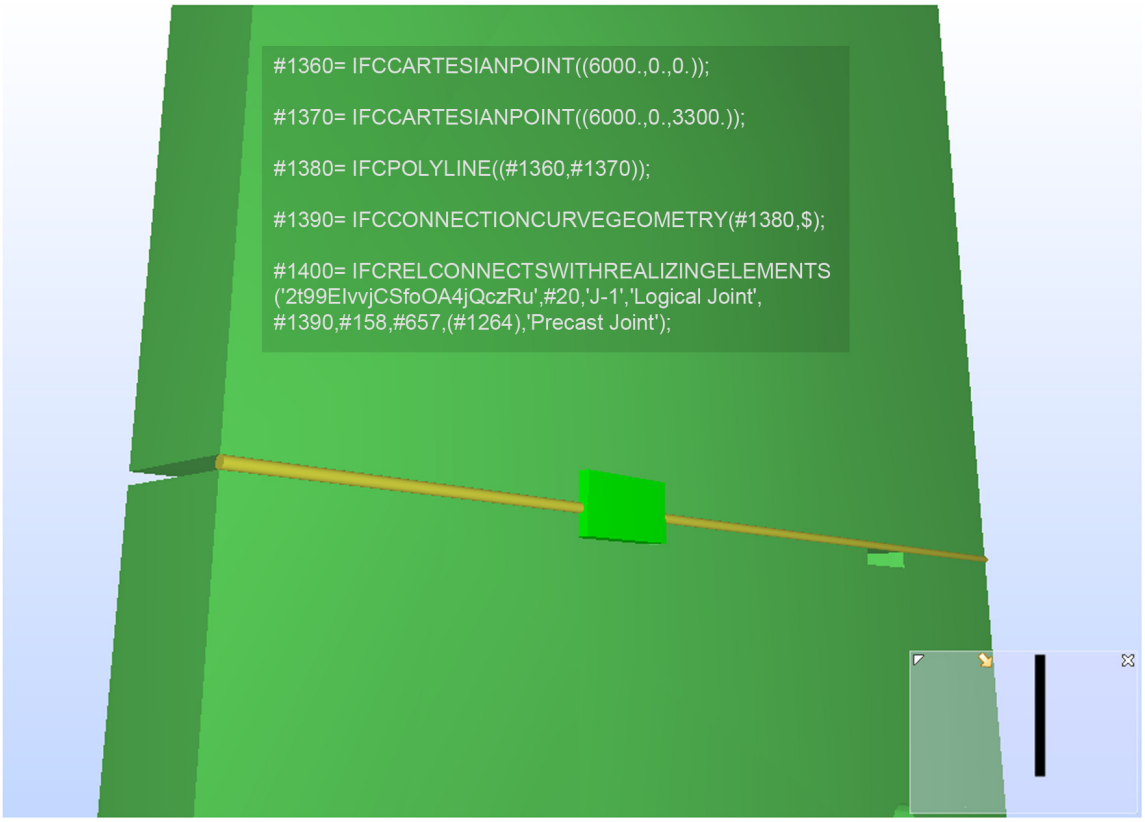

Figure 89: A test case for Precast Joint Model Exchange Validation and excerpt of the Part-21 specification [34].

for different cases ensuring good coverage of exchange models defined in Precast NBIMS jointly by the members of the PCI team.

8.4 IFC Certifications

Detailed validation plans for export and import testing are developed and proposed to the software developers. The export testing of exchange file takes place in the following manner, as illustrated in Figure 90. Initially, the MVD owner will create sample drawings with instructions for a model to convey the geometry and the specifications of the model to be built in the native application by the application owner which will be tested and exported later. In addition, the MVD owner will also provide an Express model view file that specifies the schema for the view which includes the set of entities, attributes, relationships and functions. The Express schema file is then compiled and verified for completion without any hanging references or incomplete hierarchies using an Express engine. If the express schema is successfully compiled, then an excel file containing the view definition can be verified for details about the entities, attributes and relations present, allowed values and the restrictions on them. If however, the compilation fails, then the process of Express schema file generation is repeated.

On the other hand, the application owner builds a model in the native application based on the drawings and instructions generated by the MVD owner, which needs to be tested for the required features, geometry and properties. The application owner, then generates export routines for IFC and the model is exported as an IFC file. The IFC file is then checked against the generated business rules, and a report on the test is generated. The generated test report is reviewed by the application owner and on satisfaction of all the test requirements it is approved and self certified. Any mistakes found in the export are rectified during the review process and all the processes from the creation of export routines for the IFC are repeated to generate export files satisfying all the test requirements.

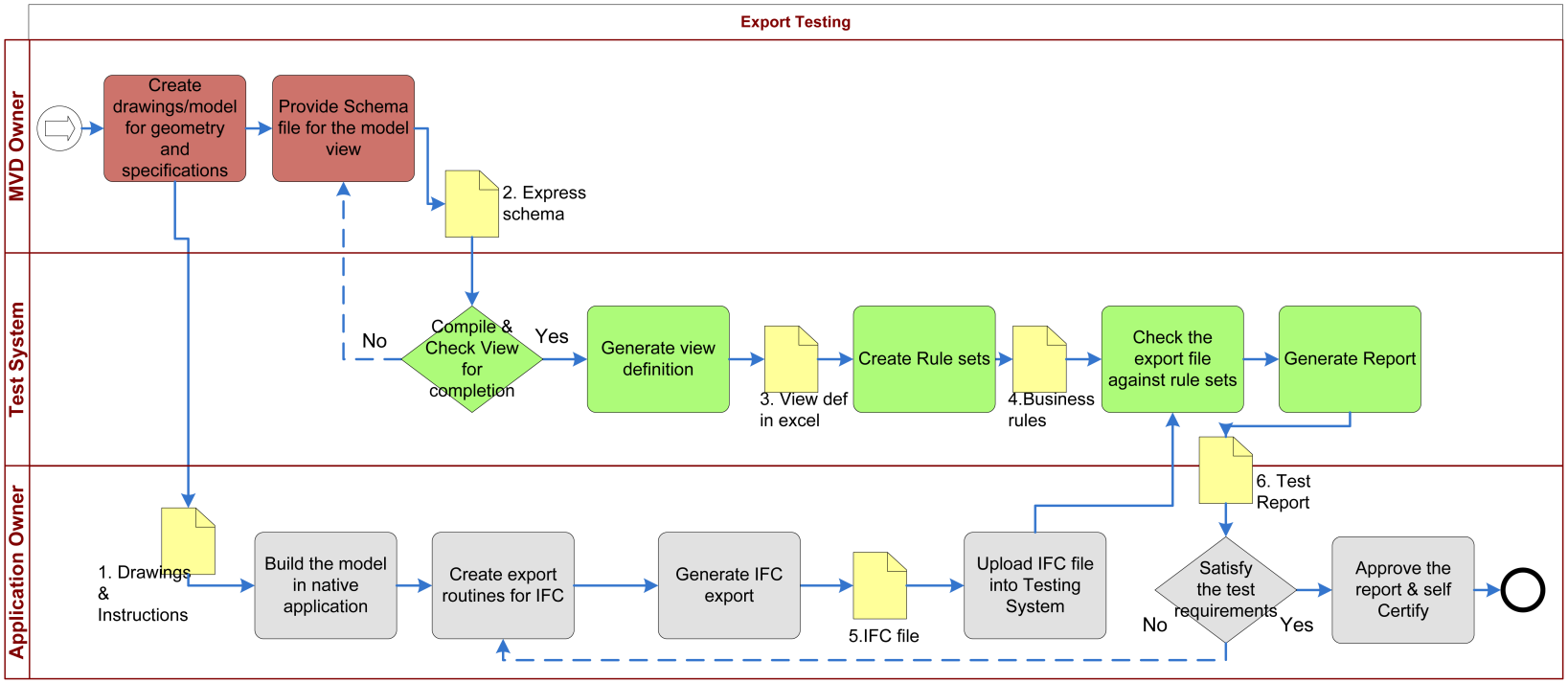

Figure 90: A process plan for export test of model exchanges.

In the case of import test, a plan is proposed as shown in Figure 91, which is a necessary test but not sufficient to satisfy import testing. The MVD owner will create subsets of view definition containing the entities, attributes, relations present and any restrictions that are imposed on them. Each of the generated subsets are to be tested/calibrated. To help in this process, test files or calibration files are generated for each of these cases and the test modules are uploaded into a test system. The application owner on the other hand creates import routines for the IFC. In addition to this the calibration files with the test modules from the test system are imported as calibration files into the native application and a report is generated with the import details. The generated report is tested to verify if it satisfies all the test requirements. If the test passes, then the test report is self certified. However, if the test fails, then the required corrections are made to the import process and the entire import process is repeated. The test system mentioned in the above sections can be any of the NBIMS certified test modules available such as the GTDS-IABI framework [65] or the N-UNIT framework provided by IFC Solutions Factory [13].

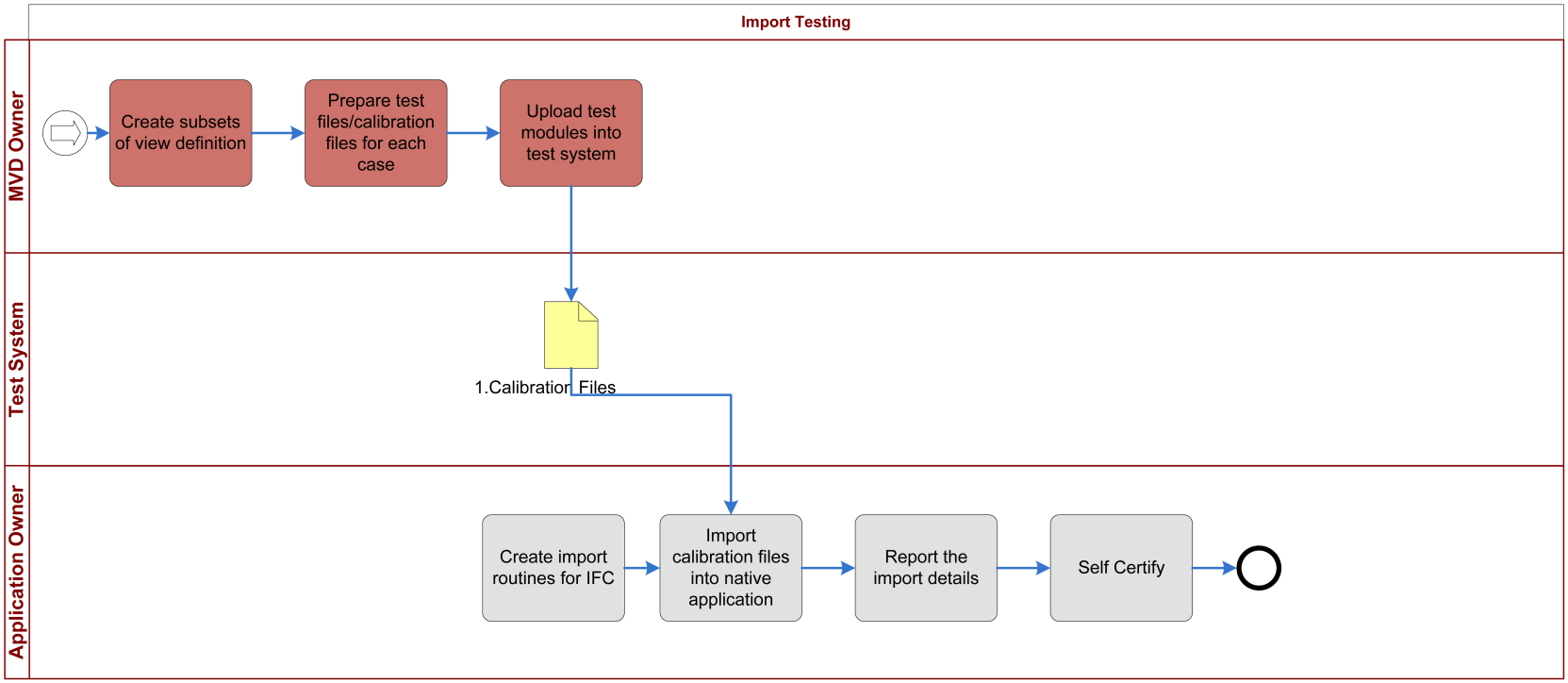

Figure 91: A process plan for import test of model exchanges (Proposed).

The model view developer plugin introduced in this chapter is envisioned to be a Web based service in the future. The features and benefits of such a system are explained below:

- BIM software users can login to a website and specify the exchange they are interested in based on SEMs
- They can submit IFC files to be validated against the MVD specification or they can use the test files provided online by authors
- The service will display what softwares have implemented the SEMs
- Users or software companies can test their exchanges and compare the results
- Public users could also rate the effectiveness of specific exchanges, record the errors, and keep version histories
- Software companies or BIM advisory teams can correct the errors and extend functionalities.

8.5 Conclusions

This chapter explained the verification and validation plans and some results. Different experiments and test cases where explored and conducted to verify and validate different aspects of the research. The evaluation metrics used are detailed in Table 9. Reasoning engines were used to verify the Ontology definitions. *Probe Classes* were made use of to introduce exceptions in the schema and to test if the system is able to handle the exceptions. This experiment proved a sufficient check for inconsistencies. Visualization of the entities in a graphical manner aided further in verifying the Ontology defined and also as a visual check for relationships.

Tests conducted on model exchanges from design packages to detailing packages (Refer Section 2.2 in this thesis) showed that IFC exports and imports are still not at an acceptable level. Semantic meaning is lost in the exchange process and needs to be recreated at the receiving end leading to additional time and cost. Detailed test cases and validation plans are defined by mapping the ontology definitions into executable C# libraries. The

idea is to show the modularization of the structure into composable units. A model view developer plugin was built in C# on top of the ontology definitions and SEM structure. This extendable plugin illustrated the workflow of completing a model exchange based on SEMs. The main advantage of this approach is that a user who is a domain expert (precast, steel, concrete, etc.) need not be familiar with IFC or model views, but can still specify a successful model exchange based only on the requirements and compile them in the form of a collection of SEMS. An elaborated scenario of developing a model view in order to automate the procurement stage of a precast project was explored using a test model that comprised of precast specific pieces such as *hollow core planks, double tee beams, precast beams, columns and spandrels,* etc. The progression of a model from a conceptual stage to the fabrication stage was outlined successfully. Details such as *connection components, corbels, lifting hooks,* etc. were also modeled. Specific test cases were run using a precast double tee, precast joint, reinforcing, etc. as examples for exchange.

A precast specific model view was compiled in EXPRESS format and validated using EXPRESS Engine. Long term validation plans are explained for the implementation of model views by software vendors and IFC certification.

CHAPTER IX

IMPACT OF RESEARCH

This chapter discusses the impact of research. The main contribution of this research is a classification structure for IFC implementations based on which a methodology to develop Model Views using Semantic Exchange Modules is introduced. This will help to improve the workflow in a number of applications in an AEC industry involving IFC product model, thereby improving the utility of IFC as a standard for interoperability. The introduction of tested and validated modular SEMs can ease the load on validation and certification of IFC implementations. Moreover, based on SEMs an automated model exchange methodology is proposed as part of future work.

9.1 Robust and Consistent approach for Model Exchange

A direct impact of the research pursued in Ontology can be felt in terms of reduction of effort and rework and an improved consistency in developing model views. An ideal workflow using ontological definitions would be as follows:

- User defines exchange model requirements.
- Requirements are mapped on the basis of SEMs.
- A new MVD is generated, based purely on the SEMs.

A model view definition, which supports an exchange requirement, can be specified solely based on the SEM packages. The exchange requirements have a direct mapping to the SEM structure (intuitive) and provide a means to develop new MVDs in a plug-and-play manner. Extensive work and time is saved by this new method. Further, the MVDs developed using this method are more consistent with each other. Since the MVD is based on SEMs and is independent of the actual implementation, the MVDs formed in this way

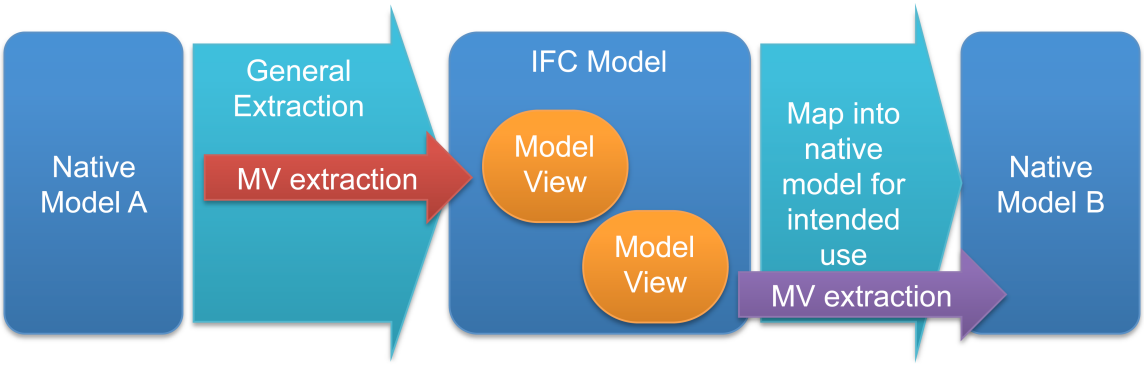

Figure 92: An illustration of model view approach using SEM as a subset of the native model on one platform that is extracted and mapped to another platform [31].

are flexible and at the same time independent of the changes to the IFC schema (schema independent). Figure 92 shows the model view approach. The capabilities of SEMs begin with the definition of a single SEM and its IFC binding. The first level of executability is the composition of concepts and the definition of a syntactically correct IFC model view. This requires a shell supporting composition and simple syntactic checking. The second level of executability is adding the required semantically equivalent bindings of a native model in some BIM platform to the IFC bindings. Given these two equivalent structures, the mappings between them can be implemented, in one or both directions.

9.2 Testing Validation and Certification of IFC Implementation

Formalizing IFC semantics will help new business capabilities both directly and indirectly. At the same time, it will also reduce the R&D expenditure of small and medium software companies by minimizing rework and re-implementation of IFC import and export functions. Embedding semantics is a unique way of specifying data and data relationships and is well documented in the semantic web industry. Research has shown that maintaining status quo is a risky option as it might lead to complexity explosion.

Testing, validation and certification of new IFC implementations can benefit from this research. The current approach adopted by buildingSMART is to test each application for import, and export certification by a third party for each model view. For example, Table 15 shows the sample costs for certification of coordination view. The total cost shown is

Table 15: Coordination view certification cost table [93].

Certification Cost	
Cost Items	**Unit Price (USD)**
A. Coordination View	
bSI contribution (10percent)	1300,00
Certification center	5200,00
Maintenance 1st year	
Maintenance from 2nd year	1300,00
Export Certification (1st round)	4000,00
Import Certification (1st round)	4000,00
Coordination View total sum (year 1)	**15600,00**

only for one tool. If a company has multiple tools then they need to pay for each additional tool. Consider the case where an application supports multiple exchanges such as BIM Architectural, MEP, Structural, etc. Each of these exchanges requires certification for every version, leading to higher costs. Table 16 lists the participants of this certification process, illustrating that this is the common practice in industry leading to additional burden for BIM software developers. This table lists the software developers and the corresponding tools for which they are getting certified for and the domain they cater to and whether it is for import or export of model data.

Based on the idea of ontological definitions and IFC concepts the workflow envisioned will be as follows:

- User can upload an IFC export from any target system for validation check.
- The IFC file is parsed and checked against the SEM requirements using custom algorithms.

The requirements for validation of export files is a work-in-progress and open for discussion. The workflow overlaps with some of the previous steps and can be defined as follows:

- SEMs are first generated by packaging entities together as a module on a semantic basis (ontological mapping of the IFC entities provide strict semantic definitions and are used to avoid inconsistencies and misrepresentations).

Table 16: Participants of the official buildingSMART IFC2x3 Coordination View V2.0 certification process [93].

Software Developer	**Software Application**	**Exchange Requirement**	**Export/Import**
Archimen	Active 3D	All	Import
Autodesk	AutoCAD Architecture	Architecture	Import & Export
Autodesk	AutoCAD MEP	Building Services	Export
Autodesk	Revit Architecture	Architecture	Import & Export
Bentley Systems	Bentley Architecture	Architecture	Import & Export
Cad-Quality	CADiE Sahakka	Building Services	Import
Data Design System	DDS-CAD MEP	Building Services	Export
Design Data	SDS/2	Structural	Import & Export
Gehry Technologies	Digital Project	Architecture	Import & Export
Graphisoft	ArchiCAD	Architecture	Import & Export
International Training Institute (ITI)	Benchmark	Building Services	Export
Nemetchek All Plan	Allplan	Architecture	Import & Export
Nemetschek North America	Vectorworks	Architecture	Import & Export
Nemetschek Scia	Scia Engineer	Structural	Import & Export
Plancal	Nova	Building Services	Import & Export
Progman	MagiCaD	Building Services	Export
Solibri	Solibri Model Checker	All	Import
Tekla	Tekla Structures	Strucutral	Import & Export
Vizelia	Facility on line	All	Import

- Exchange Model requirements are mapped from IDM and represented as MVD on the basis of these SEMs.
- MVDs developed in a modular form as above provide computational flexibility and completeness in testing instance files.
- The MVD is loaded into the checking tool and the test is performed keeping the MVD as the scope. (This form of testing is potentially schema independent as a modular-MVD can be based on any schema and version).
- Testing will be in two levels;
  - Test files can be checked for the presence of these modules on one level and
  - The completeness and correctness of the modules present will be tested in the second level.

As a result of modular MVD and testable SEMs, the need for retesting is avoided and this will lead to substantial savings in terms of cost and time.

9.3 Automated Model Exchanges

This concept addresses the development and implementation of a new integrated framework that is expected to significantly reduce the time and effort required to implement new model views for interoperability. Research at Digital Building Lab at Georgia Tech is geared towards developing an automated model exchange methodology based on SEMs. The major contribution of this research is to define and test new software packaging and integration methods that are expected to greatly simplify model data exchange. This is realized by developing a new middleware technology that simplifies and partially automates the current onerous aspects of the task, and by making Model Views easily tailored for individual projects. Figure 93 shows a flowchart for such a methodology. Resolving the interoperability problem in design, engineering and construction will impact all areas of product engineering. It will lead to new opportunities to simplify, automate and restructure workflows, in ways that are too expensive today. We propose the terms e-design and e-engineering to identify

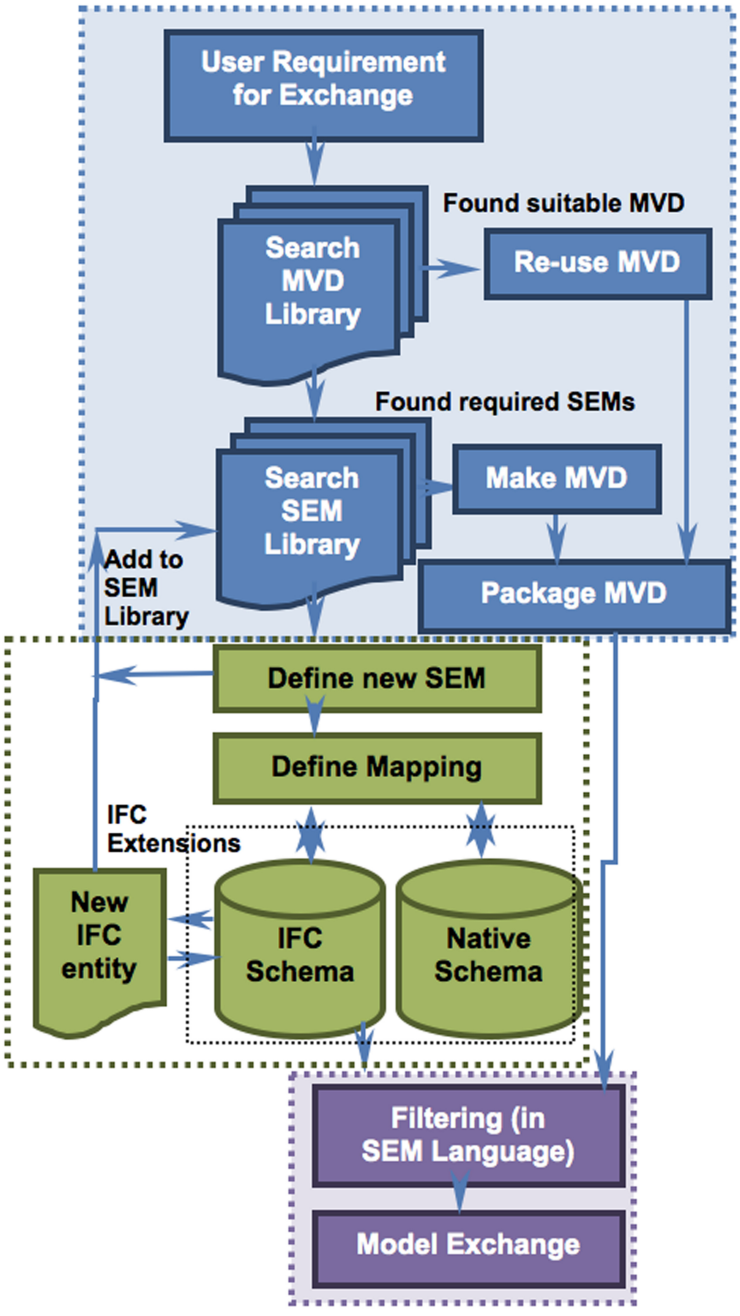

Figure 93: Automated model exchange methodology proposed based on SEM library.

the automation of regular design and engineering tasks, because reliable model exchanges will be available. Exchanges will become merged with more complex model adaptations, so that analysis/simulation interfaces can become partially or fully automated. Interactive design and linked application optimization or search are areas that can be widely expanded and utilized in everyday designs.

CHAPTER X

CONCLUSION

This chapter summarizes the results of this research and relates them to the research questions addressed. The major findings, some limitations and the future potential are explained.

1. **How can we develop model views consistently across research teams and domains?**

   In order to support IFC implementations, the consistency of model views designed is an important criteria, lack of which is causing overhead for software developers and is inhibiting new IFC implementations. Product model schemas such as IFC are rich, but redundant. Based on the insights gathered from developing the Precast National BIM Standard and further analysis as part of this research, a new methodology based on ontological definition of IFC schema called SEMs is introduced by this research. Chapter 6 explains the definition of a Precast System Ontology. Based on the analysis, it is shown that MVD development process needs to be transitioned from the current ad-hoc manner to a more rigorous framework and/or methodology similar to the one explained in this research. The semantic meaning of SEMs needs to be defined in a rigorous and formal manner with strict guidelines. This can help achieve a uniform mapping to and from internal objects of BIM tools and IFC.

2. **What should be the building blocks of model views for semantic information exchanges?**

   This research proposes defining model views based on modular, testable, and reusable packages that have an ontological definition. This definition provides a classification structure for Semantic Exchange Modules (SEM) to be developed on top of ontologies. An object oriented library of SEMs is proposed and model views are defined based on SEMs. This is explained in Chapter 7 of this thesis. SEMs, once tested

and implemented, can provide a mechanism to generate model views directly from exchange requirements. This is illustrated in Chapter 8 of this thesis through a Model View Developer Plugin, which is based on C# libraries. This is a novel idea and is yet to be fully implemented in the industry. An initial test implementation of the interoperability standards for the precast/prestressed concrete industry is illustrated in Chapter 8, with increasing levels of detail and model progression. Standard criteria for defining the SEMs proposed here should be documented to avoid various research and development teams from generating contradicting/inconsistent implementations. Such a standard approach will help in reuse of SEMs thereby resulting in the reuse of MVDs itself. This approach has the potential to reduce the current time for model view generation - implementation cycle.

This research presented the guidelines to define a SEM structure and also it's mapping to the IFC data schema. This satisfies only one branch of the SEM structure, the other branch being the mapping to the native model schema as shown in Figure 57 in Chapter 7. The mapping to the native model schema is also required to realize the full potential of the SEM notion. The implementation of mapping to native model schemas can be performed only with the support of software vendors. This is a limitation of this research currently and needs to be taken up in the future work. The implementation of mapping to native model schemas can potentially raise questions on the boundaries on which SEMs are modularized necessitating fine tuning. This concern is acknowledged and taken into consideration in the validation plans.

3. **What are the semantics of model views for information exchanges using the IFC schema?**

   Chapter 5 provides an analysis of the IFC product model schema for specific issues such as type-instancing, classification schemes, geometry, relations and rules, etc. There should be flexibility in defining the type-instance structure based on the context and nature of an application. IFC is a weak (loosely) typed system and provides multiple ways to type objects. In order to avoid ambiguities in model exchanges it is imperative that the SEMs are modeled as a strongly typed system. Such a

strongly typed SEM lattice on top of a weakly typed IFC schema can be the solution to truly realizing successful model exchanges. Classification schemes can be used to group entities and structure the data in a model exchange, thereby reducing the file size of model exchanges. This also increases the utility of the exchanged data in the importing application due to the fact that exchange already groups identical or similar objects. This is important for most BIM functionalities that involves editing or counting objects and such semantics should be specified in the model views. A multiple-inheritance structure can be the long-term solution for achieving the required flexibility in typing issues. However the study of the upward compatibility of the schema needs to be propelled by further research. This is an important research issue, to be addressed when IFC is made fully ISO compatible.

A logical framework on the basis of well-defined and unit tested SEMs, thereby following a modular approach is the future direction for creating MVDs in a standardized, and re-usable manner, cutting across all domains and providing better interoperability.

APPENDIX A

PRECAST NATIONAL BIM STANDARD

A.1 Introduction

A committee of the Precast/Prestressed Concrete Institute advised by Georgia Tech are developing a national BIM standard to support the critical information exchanges surrounding precast concrete. The precast pieces include stemmed deck members, flat deck members, beams, columns, load bearing walls and spandrels, piles, and architectural facades, plus others.

The purpose for developing a national BIM standard is to provide a layer of specificity over the top of an IFC exchange schema or other exchange schema. In the case of IFC, and possibly other schemas, the schema is highly redundant, offering many different ways to define objects, relations and attributes. The purpose of a BIM standard is to select and specify the appropriate information entities from a schema for particular uses. The selected entities that comprise a model view definition are a subset of all those in the schema.

A list of the proposed documents are provided here:

Exchange Models

http://dcom.arch.gatech.edu/pcibim/documents/processmodels/Fabrication&Erection.pdf

IDM for Precast

http://dcom.arch.gatech.edu/pcibim/documents/IDM_for_Precast.pdf

Model Views

http://dcom.arch.gatech.edu/pcibim/documents/Precast_MVDs_v2.1_Volume_I.pdf

The National Building Information Model Standard (NBIMS) is a set of interoperability standards for exchange of facility and infrastructure data throughout the life-cycle of a project. NBIMS is a joint project coordinated by the National Institute of Building Sciences (NIBS) and the buildingSMART Alliance, a working committee within NIBS, in conjunction with many other facilities-related associations and software companies worldwide.

The goal of the project within which this document was produced is to develop a national BIM standard for precast concrete design, engineering, fabrication and erection. The method of this work is to define first, the user functional requirements in a report called an Information Delivery Manuals (IDM). It was completed in January, 2009, and provided four process models that covered four distinct workflows, reflecting different project procurement methods and construction types (architectural and structural). Each of the process models was used to functionally define the information content required in each exchange.

Based on the IDM specifications, the Model View Definitions (MVD) are defined for the significant communication and data exchanges associated with all use cases associated with precast concrete. The primary orientation is that of the precast concrete fabricator. The project is funded by the Charles Pankow Foundation and the Precast/Prestressed Concrete Institute (PCI).This project was preceded by a feasibility study sponsored by the Charles Pankow Foundation to determine the issues of information exchange, focusing on architectural precast. This work has grown from and expands upon that study.

Working teams were formed within the membership of the industry BIM Advisory Committee, which functions under the auspices of the Precast/Prestressed Concrete Institute to provide the domain expertise for this endeavor. The initial step was to define the IDM use case specification. Now, the MVD are being specified and communicated with both the advisory committee for functional correctness and with software implementers.

A.2 PCI NBIMS Team

The members of each of the four working teams are listed in the following table. Their contributions are gratefully acknowledged.

PCI BIM Advisory Committee Working Teams

Chair: Michael LaNier BERGER/ABAM Engineers Inc.

1. Projects led by Precasters:
   (a) Jason Lien - chair Encon United
   (b) Jennifer Huber Encon United

(c) Monty Overstreet FDG, Inc

(d) Mike Sloter IPC Inc

(e) Mark Kraft CEG Engineers

2. Projects where Precaster is a subcontractor:

(a) Michael Slobojan chair Structureworks

(b) Mark Potter Finfrock Industries

(c) Davis Chauviere HKS Inc

(d) Charles Pool Tekla

(e) Dieter Maucher Tindall Corporation

(f) Jim Davis Stresscon

3. Architectural Precast:

(a) David Orndorff - chair Shockey Precast

(b) John Wang Mid State Precast

(c) Mike LaNier BERGER/ABAM Engineers Inc.

(d) Aaron Fink Oldcastle Precast

4. Fabrication through Erection:

(a) Mike Putich - chair

(b) Dan van Vieren Concrete Vision

(c) Wayne Norris Metromont

(d) AJ Scarfato Metromont

(e) Earle Kennett NIBS

(f) Wayne Kassian Kassian Dyck and Assoc.

(g) Karen Laptas Blue Ridge Design, Inc.

5. Technical Advisory Team:

(a) Chuck Eastman Georgia Tech

(b) Rafael Sacks Technion, Israel Institute of Technology

(c) Ivan Panushev Georgia Tech

(d) Manu Venugopal Georgia Tech

(e) Shiva Aram Georgia Tech

6. Technical Consultants:

(a) Richard See Digital Alchemy

(b) Thomas Liebich AEC3

A.3 Precast NBIMS Exchange Model Requirements

This section defines the data exchange functional requirements for exchanges between architects, engineers, general contractors, and precast fabricators, as defined by the PCI team in the Information Delivery Manual.

Precast concrete includes external cladding, structural elements, and entire building systems fabricated off-site of concrete, then erected to make up various portions of an overall project. Precast as a building system, highly interacts with many other aspects of a building. IT provides all or part of the external shell or the fundamental building structures; it must transfer its loads to the building foundations. Also, the precast pieces have multiple internal components, including pretension tendon, reinforcing, connection hardware, plus embedded components of other systems. Detailed functional descriptions of the information exchanges are provided in table 17. All of the available information items for the domain of precast concrete are organized in a hierarchy of information item groups, information items, attribute sets and attributes, which are defined as follows:

- **Information Groups** represent the major classes of objects in a building model such as site, buildings, assemblies, precast pieces, opening, rebars, etc.
- **Information Items** are specific examples of the members of each information groups. They are defined subject to the assumption that every information item in an information group has the same attributes.

- **Attribute Sets** are groups of properties that are used to describe and information group.
- **Attributes** are the properties that are needed to fully define the information group.

Table 17: Precast NBIMS Information Delivery Manual

Information Group	Information Items	Attribute Set	Attributes
Project: **Participants**	Owner, Architect of Record,	Identity	Name, Function
		Contact Info	Addresses
			Phones, email, etc.
Project: **Site**	Site	Perimeter	2D Geometry
		Location	Longitude, Latitude, Orientation
		Topography	Digital terrain model or contours
		Assembly relations	Contains buildings…
		Meta Data	Author, Version, Date
			Approval Status, Date
Building(s)	Building	Location on site	Position and orientation
		Grid geometry & control planes	Origin, directions, steps, labels
		Design constraints	Classification
			Use Occupancy
			Live Loads
			Wind Loads
			Fire Rating
			Importance Factors
			Seismic design requirements
		Structural Loads	Classification
			Use Occupancy
			Fire Rating
			Importance Factors
		Assembly relations	Located on site…
			Contains building systems…
		Association relations	Other buildings on site…
		Meta Data	Author, Version, Date
			Approval Status, Date
Spaces	Building Stories, Elevator and Core Shafts, Atriums	Geometry	Plan Outline, Heights, Voids, Elevations
			Net and gross area
			Net and gross volume
		Position	Above, below grade, % open
		Design constraints	Classification
			Use Occupancy
			Fire Rating
		Assembly relations	Located in building…
			Contains building systems…
		Meta Data	Author, Version, Date
			Approval Status, Date

Information Group	Information Items	Attribute Set	Attributes
Primary CIP or Steel Super Structure - **Assemblies**	Slab Systems, Building Cores	Geometry	Extruded shapes or solid forms
		Material	Material type
			Quantity
		Finishes	Geometry
			Surface treatments
		Assembly relations	Part of building
		Association relations	Implements structrual object..
		Nested relations	Contains components
		Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Primary CIP or Steel Super Structure - **Parts**	Beams, Columns, Bracing, Concrete Slabs, Steel Decks, Walls, Stairs, Retaining walls	Shape	Extruded shapes or solid forms
			Dimensional Tolerance Info
		Type	Structural Type (Steel, CIP Concrete)
		Supplier	GC/Contractor/ Fabricator type/name
		Material	Material type
			Quantity
		Assembly relations	Part of structural system (slab, floor, façade, frame)
		Nested relations	Contains rebars/tendons…
			Contains connection hardware….
		Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Foundations	Grade Beam, Pier Cap, Spread Footing, Slab on Grade, Stem Wall, Retaining Wall, Drilled Pier, Cassion, Pile, Pile Cap	Shape	Extruded shapes or solid forms; exposed surfaces may have different resolution
			Dimensional Tolerance Info
		Type	Structural Type (CIP Concrete/Precast)
		Supplier	GC/Contractor/Fabricator
		Material	Material type
			Quantity
		Assembly relations	Part of structural system (slab, floor, façade, frame)
		Nested relations	Contains rebars/tendons…
			Contains connection hardware….
		Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date

Information Group	Information Items	Attribute Set	Attributes
Secondary CIP or Steel Parts	Corbels, Haunches, Slab Edge, Brackets, Kickers	Shape	Extruded shapes or solid forms
		Type	Structural Type (CIP Concrete/Steel)
		Supplier	GC/Contractor/Fabricator
		Material	Material type
			Quantity
		Design loads	Moments, Reactions, Deflections
		Assembly relations	Part of structural system (slab, floor, façade, frame)
		Nested relations	Part of…concrete piece
			Contains components…rebars,
		Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Facade/Curtain Wall Assemblies	Facades	Layout Geometry	Position and orientation
			Dimensional Tolerance Info
		Grid geometry	Origin, directions, steps, reference surface, label
		Finishes	surface polygon, depth
			Surface treatments
			Materials/ Material Types
		Design loads/ constraints	Wind Loads
			Thermal performance requirements
			Acoustic performance requirements
		Assembly relations	Part of building
			Contains openings…
		Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Openings	Doors, Windows, Openings, Louvers/Grills, HVAC, Mechanical Pipe, Standpipe, Y Connection, Knox Box	Position and Geometry	Position, orientation, shape
			Location constraints
		Specification	Opening type
			Frame profile geometry
			Frame Materials/ Material Types
			Glazing/grill specifications
		Supplier	GC/Contractor/Fabricator type/name
		Assembly relations	Part of façade/curtain wall
			Contains parts…
		Meta Data	Author, Version, Date
			Approval Status, Date

Information Group	Information Items	Attribute Set	Attributes
Precast: **Structural Assemblies - Slabs**	SLABS Hollow Core, Double Tee, Flat Plank, Diaphragm	Layout	3D solid Geometry
			Dimensional Tolerance Info
		Material	Material type
			Quantity
		Topping	Outline
			Thickness / contours
			Material Type
			Surface Treatment
		Design loads	Live Loads
			Seismic Loads
			Snow Loads
			Displacements
			Wind Loads
			Thermal Loads
			Creep, Shrinkage
		Assembly relations	Belongs to… story/building
			Composed of… pieces
		Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Precast: **Non-load bearing pieces**	Architectural Spandrels (Pocket, Ledge, Button Haunch), Wall Panels, Column Covers, Mullions, Heads, Sills, Cap, Ornaments	Shape	Composed solid geometry; exposed surfaces may have higher resolution
			Gross/Net Area
			Gross/Net Volume
			Openings/Voids geometry
			Dimensional Tolerance Info
		Identification	Product Code
			Piece Mark
			Production Control Number
			Location Number
		Materials	Main Material type
			Quantity
			Concrete Mix Reference
		Insulation	Thermal/acoustic insulation type
			Wythe width and depth
			Quantity
		Finishes	Surfaqce polygon shape, depth
			Surface treatment application
			Material Type References
			Concrete Mix Reference
		Production	Status
			History
			Condition
		Design loads/ constraints	Wind Loads
			Thermal performance requirements
			Acoustic performance requirements
			Thermal loadings

Information Group	Information Items	Attribute Set	Attributes
		Assembly relations	Part of facade system...
			Contain opening parts...
		Nesting relations	Contains components…rebars,
		Structural Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Joint relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Precast: **Load-bearing pieces**	**BEAMS** (Rectangular, Inverted Tees, L Beams, Transfer Beams, Structural Spandrels (Pocket, Ledge, Button Haunch), Deep Beams) **MODULES** (Cell/Room, Core, Stair, Building) SLAB PARTS (Hollow-core, Double Tees, Flat Pieces) W**ALLS** (Core, Lite-, Shear, K-Frames, Pilasters, Insulated, Hollow-core) **STAIRS** (with integral landings, without landings) F**OUNDATIONS** (Grade Beam, Pier Cap, Spread Footing, Slab on Grade, Stem Wall, Retaining Wall, Drilled Pier, Caisson, Pile, Pile Cap)	Shape	3D composed geometry, some surface may have higher resolution
			Gross/Net Area
			Gross/Net Volume
			Openings/Voids geometry
			Dimensional Tolerance Info
		Identification	Product Code
			Piece Mark
			Production Control Number
			Location Number
		Material	Material type
			Quantity
			Concrete Mix Reference
		Finishes	surface polygon, depth
			Surface treatment application
			Material Type References
			Concrete Mix Reference
		Production	Status
			History
			Condition
		Structural Design	Applied Loads
			Moments, Shears, Reactions, Deflections
		Assembly relations	Part of structural system (slab, floor, façade, frame)
			Contain opening parts...
		Nested relations	Part of…concrete piece
			Contains components…rebars,
		Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Joint relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date

Information Group	Information Items	Attribute Set	Attributes
Precast: **Voided pieces**	Box Beams, Box Slab, Hollow Core, Bridge Segments	Shape	Hollow Profile extruded Geometry
			Gross/Net Area
			Gross/Net Volume
			Openings/Voids geometry
			Dimensional Tolerance Info
		Identification	Product Code
			Piece Mark
			Production Control Number
			Location Number
		Material	Material type
			Quantity
			Concrete Mix Reference
		Finishes	Surface polygon shape, depth
			Surface treatment application
			Material Type References
			Concrete Mix Reference
		Production	Status
			History
			Condition
		Structural Design	Applied Loads
			Moments, Shears, Reactions, Deflections
		Assembly relations	Part of structural system (slab, floor, façade, frame)
			Contain opening parts...
		Nested relations	Part of…concrete piece
			Contains components…rebars,
		Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Joint relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Other Building Systems	Glass, Store Front, Metal Cladding, EIFS, Louvers / Grill, Masonry, Architectural Steel, Electric, Security, Communication HVAC, Mechanical (Drain, Grate, Sun Shade, etc), Plumbing	Type	Assembly Type/Function
		Supplier	GC/Contractor/Fabricator
		Material	Material type
			Quantity/scope/value
		Design	Performance levels
		Assembly relations	Part of building...
			Comprises building system parts...
		Association relations	Serves spaces...
		Meta Data	Author, Version, Date
			Approval Status, Date

Information Group	Information Items	Attribute Set	Attributes
Other Building Parts	Doors and Windows, Store Front parts, Metal Cladding, EIFS, Louvers / Grills, Masonry Walls, Architectural Steel, Electric System Components, Security, Communication HVAC Ducts and ATUs,	Shape	Geometry
		Type	Function
		Supplier	GC/Contractor/Fabricator type/name
		Material	Material type
			Quantity
		Design	Performance levels
		Assembly relations	Part of building system ...
		Connection relations	.. to other system parts
			.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Precast: **Connections (Logical)**	<All precast connection types>	Location(s)	Position on each piece
		Identification	Type
			Connection Number
		Structural Design	Applied Loads
			Moments, Shears, Reactions, Deflections
		Assembly relations	Contains component parts…
		Connection relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Precast: **Plant Applied Connection Components**	**PLANT applied embeds:** Embeds (Plates, Bolts, etc.), Vendor Embed (Slotted Inserts, Threaded Inserts, etc.), Haunches, Corbel, Contractor Embeds, Masonry Clip, Dove Tail, Reglet	Shape	3D geometry, some surfaces may have different resolution
			Gross/Net Surface Area
			Gross/Net Volume/Weight
			Dimensional Tolerance Info
		Identification	Piece Mark
			Production Control Number
			Location Number
		Material	Material type
			Quantity
		Production	Status
			History
			Treatments (galvanized, plated, etc.)
			Supplier(s)
			Condition
		Structural Design	Moments, Shears, Reactions, Deflections
		Assembly relations	Part of connection…
		Nested relations	Part of…concrete piece
		Meta Data	Author, Version, Date
			Approval Status, Date

Information Group	Information Items	Attribute Set	Attributes
Precast: **Field Applied Connection Components**	**FIELD applied parts and materials:** Erection Material, Weld On, Contractor Embeds, Grout, Bearing Pad / Material, Shim Stack, Haunches, Corbel, Contractor Embeds, Masonry Clip, Dove Tail, Reglet	Shape	Geometry
			Gross/Net Surface Area
			Gross/Net Volume/Weight
			Dimensional Tolerance Info
		Identification	Piece Mark
			Production Control Number
			Location Number
		Material	Material type
			Quantity
		Production	Status
			History
			Treatments (galvanized, plated, etc.)
			Supplier(s)
			Party responsible for application
			Condition
		Structural Design	Moments, Shears, Reactions, Deflections
		Assembly relations	Part of connection…
		Nested relations	Part of…concrete piece
		Meta Data	Author, Version, Date
			Approval Status, Date
Precast: **Joints**	Grout, dry-pack, plastic profiles	Shape	Geometry (cross section shape)
			Gross/Net Length
			Dimensional Tolerance Info
		Material	Material type
			Quantity
		Finishes	Surface treatment application details
			Materials/ Material Types
		Application	Status
			History
			Condition
		Functional Requirements	Thermal/Water/Acoustic
			Expansion/Contraction
		Assembly relations	Part of structural system (slab, floor, façade, frame)
		Joint relations	.. to Precast
			.. to CIP
			.. to Steel
		Meta Data	Author, Version, Date
			Approval Status, Date
Precast: **Reinforcement**	Prestress Strand, Mesh, Welded Wire Fabric, Engineered Mesh, Rebar, Post Tension Strand,	Shape	Geometry
			Gross/Net Weight
			Shape catalog reference
			Dimensional Tolerance Info
		Identification	Rebar Mark
			Control Number

Information Group	Information Items	Attribute Set	Attributes
	Post Tension Duct, Carbon Fiber Mesh, Carbon Fiber Reinforcement	Material	Material type
			Quantity
		Production	Status
			History
			Condition
		Structural Design	Tensile/Compressive Stresses
			Shear stress
		Assembly relations	Part of structural component or connection
		Nested relations	Part of…concrete piece
		Meta Data	Author, Version, Date
			Approval Status, Date
Precast: **Lifting Devices**	Vendor Insert, Strand Loop, Frames, Knock Out panels, Embed Plate	Shape	Geometry
			Gross/Net Volume
		Identification	Piece Mark
			Control Number
		Material	Material type
			Quantity
		Production	Status
			History
			Condition
		Structural Design	Applied Loads
		Nested relations	Part of…concrete piece
		Meta Data	Author, Version, Date
			Approval Status, Date
Precast: **Concrete Mixes**	Concrete Mix Definition	Material	Material type
			Supplier
		Structural properties	Applied Stresses
			Shrinkage, Creep Requirements
			Curing rate, moisture required, workability (slump), etc.
		Other properties	Color, aesthetic properties
			Fire rating
		Meta Data	Author, Version, Date
			Approval Status, Date
Precast: **Finish Material Types**	Finish Material Definition	Material	Material type
			Surface treatment application details
		Patterns	Geometry
		Production	Status
			History
			Supplier
			Condition
		Meta Data	Author, Version, Date
			Approval Status, Date
	End	End	End

APPENDIX B

PRECAST SYSTEM ONTOLOGY SPECIFICATION IN OWL

The Precast System Ontology specification in Web Ontology Language (OWL) is presented here.

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://dcom.arch.gatech.edu/ontologies/2010/IFCOntology.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     ontologyIRI="http://dcom.arch.gatech.edu/ontologies/2010/
     IFCOntology.owl">
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="owl2xml" IRI="http://www.w3.org/2006/12/owl2-xml#"/>
    <Prefix name="" IRI="http://dcom.arch.gatech.edu/ontologies/2010
    /IFCOntology.owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Prefix name="IFCOntology"
    IRI="http://dcom.arch.gatech.edu/ontologies/2010/IFCOntology.owl#"/>
    <Annotation>
        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
        <Literal datatypeIRI="&rdf;PlainLiteral">
        An IFC ontology that describes various entities, relations and
        variables</Literal>
    </Annotation>
    <Declaration>
        <Class IRI="#Enumerations"/>
```

```
</Declaration>
<Declaration>
    <Class IRI="#Extrusion"/>
</Declaration>
<Declaration>
    <Class IRI="#Geometry"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcBuildingElement"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcBuildingElementPart"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcDiscreteAccessory"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcDistributionElement"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcElement"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcElementAssembly"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcElementComponent"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcFastener"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcFeatureElement"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcFurnishingElement"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcGeographicElement"/>
</Declaration>
<Declaration>
```

```
    <Class IRI="#IfcGridPlacement"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcLocalPlacement"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcMaterialDefinitionRepresentation"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcMechanicalFastener"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcObjectPlacement"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcProduct"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcProductDefinitionShape"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcProductRepresentation"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcReinforcingBar"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcReinforcingElement"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcReinforcingMesh"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcTendon"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcTendonAnchor"/>
</Declaration>
<Declaration>
    <Class IRI="#IfcTransportElement"/>
</Declaration>
```

```
<Declaration>
    <Class IRI="#IfcVirtualElement"/>
</Declaration>
<Declaration>
    <Class IRI="#Profile"/>
</Declaration>
<Declaration>
    <Class IRI="#Solid"/>
</Declaration>
<Declaration>
    <Class IRI="#ValuePartition"/>
</Declaration>
<Declaration>
    <Class IRI="#piece"/>
</Declaration>
<Declaration>
    <Class abbreviatedIRI="owl:Thing"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#hasGeometry"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#objectPlacement"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#placesObject"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#representation"/>
</Declaration>
<EquivalentClasses>
    <Class IRI="#Geometry"/>
    <ObjectUnionOf>
        <Class IRI="#Extrusion"/>
        <Class IRI="#Profile"/>
        <Class IRI="#Solid"/>
    </ObjectUnionOf>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#IfcElementComponent"/>
    <ObjectIntersectionOf>
```

```
        <Class IRI="#IfcElement"/>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#objectPlacement"/>
            <Class IRI="#IfcObjectPlacement"/>
        </ObjectSomeValuesFrom>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#representation"/>
            <Class IRI="#IfcProductRepresentation"/>
        </ObjectSomeValuesFrom>
        <ObjectAllValuesFrom>
            <ObjectProperty IRI="#representation"/>
            <Class IRI="#IfcProductDefinitionShape"/>
        </ObjectAllValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
<SubClassOf>
    <Class IRI="#Extrusion"/>
    <Class IRI="#Geometry"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Geometry"/>
    <Class IRI="#ValuePartition"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcBuildingElement"/>
    <Class IRI="#IfcElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcBuildingElementPart"/>
    <Class IRI="#IfcElementComponent"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcDiscreteAccessory"/>
    <Class IRI="#IfcElementComponent"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcDistributionElement"/>
    <Class IRI="#IfcElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcElement"/>
```

```
    <Class IRI="#IfcProduct"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcElement"/>
    <ObjectUnionOf>
        <Class IRI="#IfcBuildingElement"/>
        <Class IRI="#IfcDistributionElement"/>
        <Class IRI="#IfcElementAssembly"/>
        <Class IRI="#IfcFeatureElement"/>
        <Class IRI="#IfcFurnishingElement"/>
        <Class IRI="#IfcGeographicElement"/>
        <Class IRI="#IfcTransportElement"/>
        <Class IRI="#IfcVirtualElement"/>
    </ObjectUnionOf>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcElementAssembly"/>
    <Class IRI="#IfcElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcFastener"/>
    <Class IRI="#IfcElementComponent"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcFeatureElement"/>
    <Class IRI="#IfcElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcFurnishingElement"/>
    <Class IRI="#IfcElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcGeographicElement"/>
    <Class IRI="#IfcElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcGridPlacement"/>
    <Class IRI="#IfcObjectPlacement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcLocalPlacement"/>
```

```
    <Class IRI="#IfcObjectPlacement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcMaterialDefinitionRepresentation"/>
    <Class IRI="#IfcProductRepresentation"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcMechanicalFastener"/>
    <Class IRI="#IfcElementComponent"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcProduct"/>
    <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#objectPlacement"/>
        <Class IRI="#IfcObjectPlacement"/>
    </ObjectSomeValuesFrom>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcProduct"/>
    <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#representation"/>
        <Class IRI="#IfcProductRepresentation"/>
    </ObjectSomeValuesFrom>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcProductDefinitionShape"/>
    <Class IRI="#IfcProductRepresentation"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcProductDefinitionShape"/>
    <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#hasGeometry"/>
        <Class IRI="#Geometry"/>
    </ObjectSomeValuesFrom>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcReinforcingBar"/>
    <Class IRI="#IfcReinforcingElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcReinforcingBar"/>
```

```
    <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcReinforcingBar"/>
    <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#representation"/>
        <Class IRI="#IfcProductDefinitionShape"/>
    </ObjectSomeValuesFrom>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcReinforcingBar"/>
    <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#representation"/>
        <ObjectIntersectionOf>
            <Class IRI="#IfcProductDefinitionShape"/>
            <ObjectSomeValuesFrom>
                <ObjectProperty IRI="#hasGeometry"/>
                <Class IRI="#Extrusion"/>
            </ObjectSomeValuesFrom>
        </ObjectIntersectionOf>
    </ObjectSomeValuesFrom>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcReinforcingElement"/>
    <Class IRI="#IfcElementComponent"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcReinforcingElement"/>
    <ObjectAllValuesFrom>
        <ObjectProperty IRI="#representation"/>
        <Class IRI="#IfcProductDefinitionShape"/>
    </ObjectAllValuesFrom>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcReinforcingMesh"/>
    <Class IRI="#IfcReinforcingElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcTendon"/>
    <Class IRI="#IfcReinforcingElement"/>
</SubClassOf>
```

```
<SubClassOf>
    <Class IRI="#IfcTendonAnchor"/>
    <Class IRI="#IfcReinforcingElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcTransportElement"/>
    <Class IRI="#IfcElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#IfcVirtualElement"/>
    <Class IRI="#IfcElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Profile"/>
    <Class IRI="#Geometry"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Solid"/>
    <Class IRI="#Geometry"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#piece"/>
    <Class IRI="#IfcBuildingElement"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#piece"/>
    <Class IRI="#IfcFastener"/>
</SubClassOf>
<DisjointClasses>
    <Class IRI="#Extrusion"/>
    <Class IRI="#Profile"/>
    <Class IRI="#Solid"/>
</DisjointClasses>
<DisjointClasses>
    <Class IRI="#IfcBuildingElement"/>
    <Class IRI="#IfcDistributionElement"/>
    <Class IRI="#IfcElementAssembly"/>
    <Class IRI="#IfcFeatureElement"/>
    <Class IRI="#IfcFurnishingElement"/>
    <Class IRI="#IfcGeographicElement"/>
    <Class IRI="#IfcTransportElement"/>
```

```
    <Class IRI="#IfcVirtualElement"/>
</DisjointClasses>
<DisjointClasses>
    <Class IRI="#IfcBuildingElement"/>
    <Class IRI="#IfcElementComponent"/>
</DisjointClasses>
<DisjointClasses>
    <Class IRI="#IfcBuildingElementPart"/>
    <Class IRI="#IfcDiscreteAccessory"/>
    <Class IRI="#IfcFastener"/>
    <Class IRI="#IfcMechanicalFastener"/>
</DisjointClasses>
<DisjointClasses>
    <Class IRI="#IfcGridPlacement"/>
    <Class IRI="#IfcLocalPlacement"/>
</DisjointClasses>
<DisjointClasses>
    <Class IRI="#IfcMaterialDefinitionRepresentation"/>
    <Class IRI="#IfcProductDefinitionShape"/>
</DisjointClasses>
<InverseObjectProperties>
    <ObjectProperty IRI="#objectPlacement"/>
    <ObjectProperty IRI="#placesObject"/>
</InverseObjectProperties>
<FunctionalObjectProperty>
    <ObjectProperty IRI="#hasGeometry"/>
</FunctionalObjectProperty>
<FunctionalObjectProperty>
    <ObjectProperty IRI="#objectPlacement"/>
</FunctionalObjectProperty>
<ObjectPropertyRange>
    <ObjectProperty IRI="#hasGeometry"/>
    <Class IRI="#Geometry"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#objectPlacement"/>
    <Class IRI="#IfcObjectPlacement"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#placesObject"/>
    <Class IRI="#IfcProduct"/>
```

```
    </ObjectPropertyRange>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
        <IRI>#IfcObjectPlacement</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">Abstract supertype for the
        special types defining the object coordinate system.
        The IfcObjectPlacement has to be provided for each product that
        has a shape representation.</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
        <IRI>#IfcProduct</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">Any Object that relates to a
        geometric or spatial context. Subtypes of IfcProduct usually hold a
        shape representation and a object placement within the project
        structure.</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
        <IRI>#IfcProductRepresentation</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">Defines a representation
        of a product, including its (geometric or topological) representation.
        A product can have zero, one or many geometric representations,
        and a single geometric representation can be shared among various
        products using mapped representations.</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
        <IRI>#IfcReinforcingElement</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">Bars, wirs, strands, and
        other slender members embedded in concrete in such a manner
        that the reinforcement and the concrete act together in resisting
        forces.</Literal>
    </AnnotationAssertion>
</Ontology>
```

APPENDIX C

TEST MODEL FOR PRECAST MODEL VIEW VALIDATION

A test model was created for the validation of the precast model view. The model is comprised of a structure that is two stories in height. It is erected in the form of two slab bays; this makes four slabs total and 6 columns. Slabs on floor 1 are Hollowcore (HC) planks and slabs on 2nd floor are made up of Doubletee (DT) beams. The column spacing is fixed at 40 ft. by 50 ft. All beams and spandrels supported on columns by corbels. Both stories of slabs have a topping above DT and HC. Figure 94 shows the illustration of the IFC file in Solibri Model Viewer. The test file in **.ifc** *format is presented here. This is only an excerpt showing the important features, the full model will be provided upon request.*

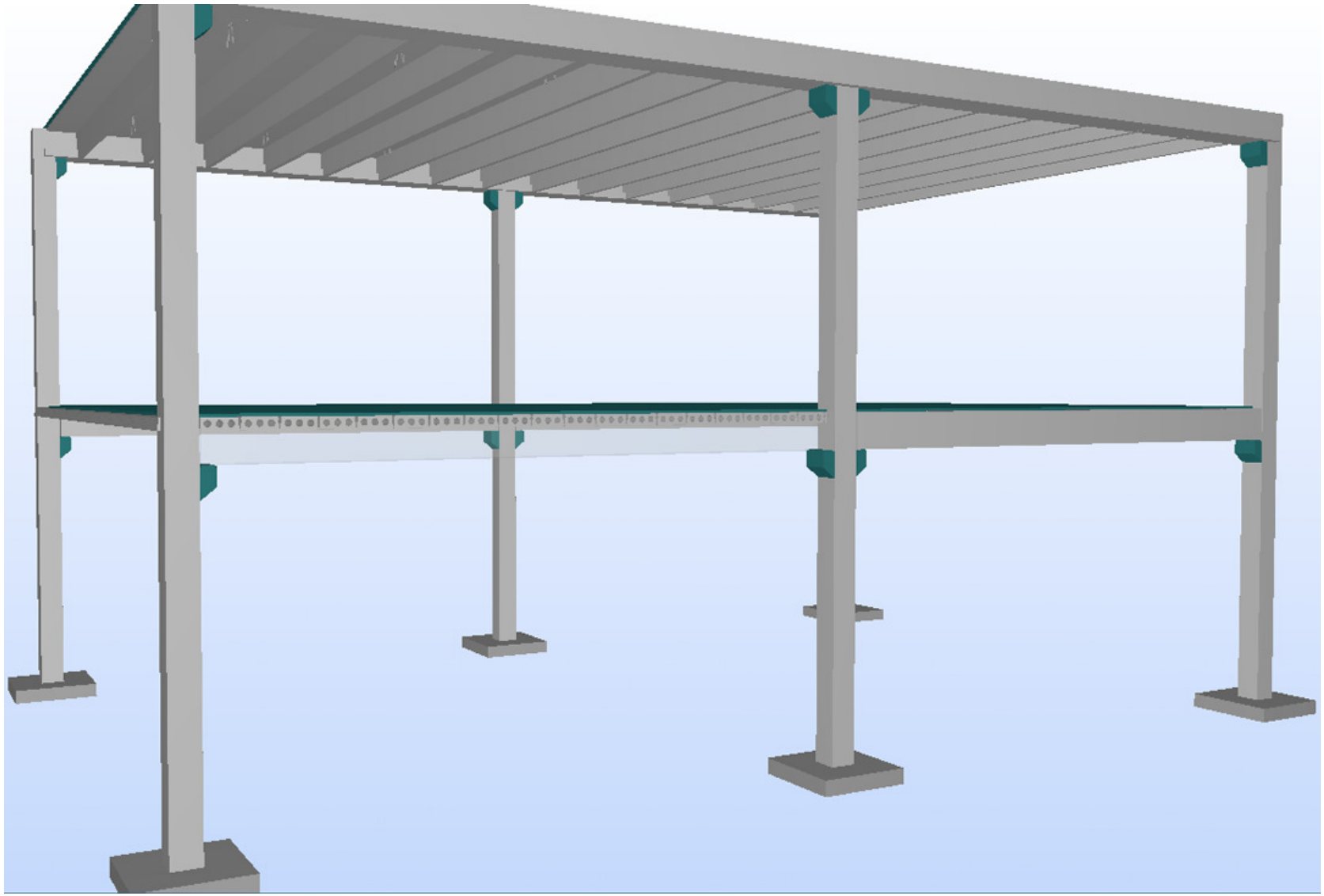

Figure 94: Test model for precast model view validation.

```
/*Test model begins here*/
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((’ViewDefinition [CoordinationView, QuantityTakeOffAddOnView]’),’2;1’);
FILE_NAME(’C:\\TeklaStructuresModels\\GT_NBIMS_TestModel\\GT_NBIMS_TestModel_v1.3.ifc’,
’2011-08-29T12:23:55’,(’AD/mvenugopal3’),(’Structural Designer’),’EXPRESS Data Manager version:20070116’,
’Tekla Structures 16.1 Build:624002/31.8.2010, IFC Export Version:136/May  3 2010’,’’);
FILE_SCHEMA((’IFC2X3’));
ENDSEC;

DATA;
#1= IFCPERSON(’AD/mvenugopal3’,’Undefined’,$,$,$,$,$,$);
#3= IFCORGANIZATION($,’Tekla Corporation’,$,$,$);
#7= IFCPERSONANDORGANIZATION(#1,#3,$);
#8= IFCAPPLICATION(#3,’16.1’,’Tekla Structures’,’Multi material modeling’);
#9= IFCOWNERHISTORY(#7,#8,$,.NOCHANGE.,$,$,$,1314635018);
#10= IFCCARTESIANPOINT((0.,0.,0.));
#14= IFCDIRECTION((1.,0.,0.));
#18= IFCDIRECTION((0.,1.,0.));
#22= IFCDIRECTION((0.,0.,1.));
#26= IFCAXIS2PLACEMENT3D(#10,#22,#14);
#29= IFCGEOMETRICREPRESENTATIONCONTEXT(’Body’,’Model’,3,1.0000000E-5,#26,$);
#32= IFCGEOMETRICREPRESENTATIONCONTEXT(’BoundingBox’,’Model’,3,1.0000000E-5,#26,$);
#35= IFCSIUNIT(*,.LENGTHUNIT.,.MILLI.,.METRE.);
#36= IFCSIUNIT(*,.AREAUNIT.,$,.SQUARE_METRE.);
#37= IFCSIUNIT(*,.VOLUMEUNIT.,$,.CUBIC_METRE.);
#38= IFCSIUNIT(*,.MASSUNIT.,.KILO.,.GRAM.);
#39= IFCSIUNIT(*,.TIMEUNIT.,$,.SECOND.);
#40= IFCSIUNIT(*,.PLANEANGLEUNIT.,$,.RADIAN.);
#41= IFCSIUNIT(*,.SOLIDANGLEUNIT.,$,.STERADIAN.);
#42= IFCSIUNIT(*,.THERMODYNAMICTEMPERATUREUNIT.,$,.DEGREE_CELSIUS.);
#43= IFCSIUNIT(*,.LUMINOUSINTENSITYUNIT.,$,.LUMEN.);
#44= IFCUNITASSIGNMENT((#35,#36,#37,#38,#39,#40,#41,#42,#43));
---
/* SPATIAL HIERARCHY */

#46= IFCPROJECT(’2AGjCd8lv0QeFjaYmjHB_9’,#9,’GT_NBIMS_TestModel’,’Description’,’Object type’,
’LongName’,’Phase’,(#29,#32),#44);
#53= IFCLOCALPLACEMENT($,#26);
#56= IFCSITE(’0lfU4fsyT1NuAg_kgaD$Jb’,#9,’Undefined’,$,$,#53,$,$,.ELEMENT.,$,$,0.,$,$);
#66= IFCLOCALPLACEMENT(#53,#26);
```

```
#69= IFCBUILDING('3TCnZtKq581gGDwUq2nbhb',#9,'GT_Lab',$,$,#66,$,$,.ELEMENT.,$,$,$);
#79= IFCLOCALPLACEMENT(#66,#26);
#82= IFCBUILDINGSTOREY('3Jyan7D4bCyf0GcbjvoKxQ',#9,'Second Floor',$,$,#79,$,$,.ELEMENT.,$);

---
/*PRECAST DOUBLE TEE */

#176= IFCPOLYLINE((#112,#116,#120,#124,#128,#132,#136,#140,#144,#148,#152,#156,#160,#164,
#168,#172,#112));
#180= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'8DT18-4',#176);
#181= IFCDIRECTION((0.,-1.,0.));
#185= IFCCARTESIANPOINT((15240.,0.,0.));
#189= IFCAXIS2PLACEMENT3D(#185,#96,#181);
#192= IFCEXTRUDEDAREASOLID(#180,#189,#22,15240.);
#195= IFCSTYLEDITEM(#192,(#110),'Name');
#199= IFCSHAPEREPRESENTATION(#29,'Body','SweptSolid',(#192));
#205= IFCPRODUCTDEFINITIONSHAPE('','',(#199));
#209= IFCBEAM('1EMumx000AWp4pCJGsCZKo',#9,'PRECAST_DOUBLE_TEE','8DT18-4','8DT18-4',#103,
#205,'TS_170165');
#228= IFCBEAMTYPE('2bxBAEn3bEVvYdz11ewv5X',#9,'8DT18-4',$,$,$,$,$,$,.NOTDEFINED.);
---
/*BLOCKOUT*/

#503= IFCPOLYLINE((#471,#475,#479,#483,#487,#491,#495,#499,#471));
#507= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'',#503);
#508= IFCCARTESIANPOINT((0.,9448.8,-101.60001));
#512= IFCAXIS2PLACEMENT3D(#508,#181,#14);
#515= IFCEXTRUDEDAREASOLID(#507,#512,#22,12192.);
#518= IFCSHAPEREPRESENTATION(#29,'Body','SweptSolid',(#515));
#524= IFCPRODUCTDEFINITIONSHAPE('','',(#518));
#528= IFCOPENINGELEMENT('1EMumx000AY34pCJGsCZKo',#9,'','','Recess',#468,#524,'');
#549= IFCRELVOIDSELEMENT('0JLxRyUWvD$xPrbUmhOAZY',#9,'','',#209,#528);
---
/* PRECAST CONCRETE DOUBLE TEE*/

#837= IFCAXIS2PLACEMENT3D(#92,#22,#96);
#840= IFCLOCALPLACEMENT(#79,#837);
#843= IFCCARTESIANPOINT((0.,0.));
#847= IFCDIRECTION((1.,0.));
#851= IFCAXIS2PLACEMENT2D(#843,#847);
#854= IFCRECTANGLEPROFILEDEF(.AREA.,'',#851,508.,508.);
```

```
#855= IFCDIRECTION((0.,0.,-1.));
#859= IFCCARTESIANPOINT((15240.,-1524.,228.6));
#863= IFCAXIS2PLACEMENT3D(#859,#855,#18);
#866= IFCEXTRUDEDAREASOLID(#854,#863,#22,6705.6);
#869= IFCSHAPEREPRESENTATION(#29,'Body','SweptSolid',(#866));
#875= IFCPRODUCTDEFINITIONSHAPE('','',(#869));
#879= IFCOPENINGELEMENT('1EMumx000Ab34pCJGsCZKo',#9,'','','Recess',#840,#875,'');
#900= IFCRELVOIDSELEMENT('3XQzlrzfjD0uH2NjRSbxJ6',#9,'','',#209,#879);
#901= IFCCARTESIANPOINT((15240.,10966.45,12573.));
#905= IFCAXIS2PLACEMENT3D(#901,#22,#96);
#908= IFCLOCALPLACEMENT(#79,#905);
#911= IFCPOLYLINE((#112,#116,#120,#124,#128,#132,#136,#140,#144,#148,#152,#156,#160,
#164,#168,#172,#112));
#915= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'8DT18-4',#911);
#916= IFCAXIS2PLACEMENT3D(#185,#96,#181);
#919= IFCEXTRUDEDAREASOLID(#915,#916,#22,15240.);
#922= IFCSTYLEDITEM(#919,(#110),'Name');
#926= IFCSHAPEREPRESENTATION(#29,'Body','SweptSolid',(#919));
#932= IFCPRODUCTDEFINITIONSHAPE('','',(#926));
#936= IFCBEAM('1EMumx00078J4pCJGsCZGv',#9,'PRECAST_DOUBLE_TEE','8DT18-4','8DT18-4',
#908,#932,'TS_164018');
--
/* PRECAST BEAM */

#1627= IFCCARTESIANPOINT((15240.,12192.,5842.));
#1631= IFCAXIS2PLACEMENT3D(#1627,#22,#181);
#1634= IFCLOCALPLACEMENT(#1328,#1631);
#1637= IFCPOLYLINE((#1365,#1369,#1373,#1377,#1381,#1385,#1365));
#1641= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'L-12*24-12',#1637);
#1642= IFCCARTESIANPOINT((11912.6,0.,0.));
#1646= IFCAXIS2PLACEMENT3D(#1642,#96,#181);
#1649= IFCEXTRUDEDAREASOLID(#1641,#1646,#22,11633.2);
#1652= IFCSTYLEDITEM(#1649,(#1363),'Name');
#1656= IFCSHAPEREPRESENTATION(#29,'Body','SweptSolid',(#1649));
#1662= IFCPRODUCTDEFINITIONSHAPE('','',(#1656));
#1666= IFCBEAM('1EMEXR000S4J4pCJGqDpOs',#9,'PRECAST_BEAM','L-12*24-12','L-12*24-12',
#1634,#1662,'TS_46915');
---
/* BLOCKOUT */

#11712= IFCAXIS2PLACEMENT3D(#1940,#22,#18);
```

```
#11715= IFCLOCALPLACEMENT(#1328,#11712);
#11718= IFCAXIS2PLACEMENT2D(#843,#847);
#11721= IFCRECTANGLEPROFILEDEF(.AREA.,'',#11718,508.,508.);
#11722= IFCAXIS2PLACEMENT3D(#2502,#855,#96);
#11725= IFCEXTRUDEDAREASOLID(#11721,#11722,#22,6705.6);
#11728= IFCSHAPEREPRESENTATION(#29,'Body','SweptSolid',(#11725));
#11734= IFCPRODUCTDEFINITIONSHAPE('','',(#11728));
#11738= IFCOPENINGELEMENT('1EMEXR000RCJ4pCJGqDpOr',#9,'','','Recess',#11715,#11734,'');
#11759= IFCRELVOIDSELEMENT('1Lm_NQmcb9qArEvhHY1bg9',#9,'','',#1975,#11738);
---
/* PRECAST HOLLOWCORE */

#161600= IFCFACETEDBREP(#161596);
#161603= IFCSTYLEDITEM(#161600,(#110),'Name');
#161607= IFCSHAPEREPRESENTATION(#29,'Body','Brep',(#161600));
#161613= IFCPRODUCTDEFINITIONSHAPE('','',(#161607));
#161617= IFCBEAM('1EKgNT0001gp4pCJGmD3Gm',#9,'HC_PLANK','2LHC8','2LHC8',#158810,
#161613,'TS_3596');
#161636= IFCBEAMTYPE('0EQSUkpZ58d8$p6EmOFp8z',#9,'2LHC8',$,$,$,$,$,$,.NOTDEFINED.);
--
/* PRECAST COLUMN */

#168610= IFCAXIS2PLACEMENT3D(#168337,#22,#14);
#168613= IFCLOCALPLACEMENT(#156078,#168610);
#168616= IFCSHAPEREPRESENTATION(#29,'Body','Brep',(#168575));
#168622= IFCPRODUCTDEFINITIONSHAPE('','',(#168616));
#168626= IFCOPENINGELEMENT('1EMEXR001qlZ4pCJGqDpSt',#9,'','','Recess',#168613,#168622,'');
#168647= IFCRELVOIDSELEMENT('03CqhUEhH74PHHRA3_xNFL',#9,'','',#168371,#168626);
#168648= IFCCARTESIANPOINT((15240.,12192.,-609.6));
#168652= IFCAXIS2PLACEMENT3D(#168648,#22,#14);
#168655= IFCLOCALPLACEMENT(#156078,#168652);
#168658= IFCAXIS2PLACEMENT2D(#843,#847);
#168661= IFCRECTANGLEPROFILEDEF(.AREA.,'20"X20"',#168658,508.,508.);
#168662= IFCAXIS2PLACEMENT3D(#156111,#855,#181);
#168665= IFCEXTRUDEDAREASOLID(#168661,#168662,#22,6705.6);
#168668= IFCSTYLEDITEM(#168665,(#156105),'Name');
#168672= IFCSHAPEREPRESENTATION(#29,'Body','SweptSolid',(#168665));
#168678= IFCPRODUCTDEFINITIONSHAPE('','',(#168672));
#168682= IFCCOLUMN('1EKgNT0000K34pCJGmD38s',#9,'PRECAST_COLUMN','20"X20"','20"X20"',#168655,#168678,'TS_1907');
```

REFERENCES

[1] Adachi, Y., "Overview of IFC model server framework," in *EWork and eBusiness in architecture, engineering and construction: proceedings of the fourth European Conference on Product and Process Modelling in the Building and Related Industries, Portorož, Slovenia, 9-11 September 2002*, p. 367, Taylor & Francis, 2002.

[2] Angele, J. and Lausen, G., "Ontologies in f-logic," *Handbook on Ontologies*, pp. 29–50, 2004.

[3] Aram, V., Eastman, C., Sacks, R., Panushev, I., and Venugopal, M., "Introducing a New Methodology to Develop The Information Delivery Manual For AEC Projects," in *Proceedings of the CIB W78 2010: 27th International Conference – Cairo, Egypt, 16-18 November*, 2010.

[4] Arplrez, J., Corcho, O., Fernandez-Lopez, M., and Gomez-Perez, A., "Webode in a nutshell," *AI magazine*, vol. 24, no. 3, p. 37, 2003.

[5] Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., and Nardi, D., *The description logic handbook: theory, implementation, and applications*. Cambridge Univ Pr, 2003.

[6] Barak, R., Jeong, Y., Sacks, R., and Eastman, C., "Unique requirements of building information modeling for cast-in-place reinforced concrete," *Journal of Computing in Civil Engineering*, vol. 23, no. 2, pp. 64–74, 2009.

[7] Bazjanac, V., "IFC BIM-based methodology for semi-automated building energy performance simulation," Lawrence Berkeley National Laboratory: Lawrence Berkeley National Laboratory. LBNL Paper LBNL-919E. Retrieved from: http://escholarship.org/uc/item/0m8238pj, 2008.

[8] Bazjanac, V. and Kiviniemi, A., "Reduction, simplification, translation and interpretation in the exchange of model data," in *CIB W*, vol. 78, pp. 163–168, 2007.

[9] Bechhofer, S., Horrocks, I., Goble, C., and Stevens, R., "Oiled: a reason-able ontology editor for the semantic web," *KI 2001: Advances in Artificial Intelligence*, pp. 396–408, 2001.

[10] Beetz, J., De Vries, B., and Van Leeuwen, J., "RDF-based distributed functional part specifications for the facilitation of service-based architectures," in *Proceedings of the 24th CIB-W78 Conference on Information Technology in Construction*, Citeseer, 2007.

[11] Beetz, J., Van Leeuwen, J., and De Vries, B., "IfcOWL: A case of transforming EXPRESS schemas into ontologies," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 23, no. 01, pp. 89–101, 2009.

[12] Berners Lee, T., "Semantic web," 2002. w3c.org (last accessed 10/23/2011).

[13] Blis-Project, "IFC Solutions Factory: Model View Definitions Site," 2010. http://www.blis-project.org/IAI-MVD/ (last accessed 10/23/2011).

[14] Boehm, B., "Software engineering economics," *Software Engineering, IEEE Transactions on*, no. 1, pp. 4–21, 1984.

[15] Böhms, M., Bonsma, P., Willems, P., Zarli, A., Bourdeau, M., Pascual, E., Storer, G., Kazi, S., Hannus, M., Sedano, J., and others, "The SWOP Semantic Product Modelling Approach," Technical Report STRP NMP2-CT-2005-016972 "SWOP", TNO, 04 2008.

[16] Borgo, S. and Leitao, P., "The role of foundational ontologies in manufacturing domain applications," in *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA and ODBASE. OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2004. Proceedings, Part I, 25-29 Oct. 2004*, (Berlin, Germany), pp. 670–88, Springer-Verlag, 2004.

[17] Borrmann, A. and Rank, E., "Specification and implementation of directional operators in a 3D spatial query language for building information models," *Advanced Engineering Informatics*, vol. 23, no. 1, pp. 32–44, 2009.

[18] Borrmann, A. and Rank, E., "Topological analysis of 3D building models using a spatial query language," *Advanced Engineering Informatics*, vol. 23, no. 4, pp. 370–385, 2009.

[19] Borst, W., *Construction of engineering ontologies for knowledge sharing and reuse.* PhD thesis, Centre of Telematica and Information Technology, Universiteit Twente: Enschede, The Netherlands, 1997.

[20] Box, G. and Draper, N., *Empirical model-building and response surfaces.* John Wiley & Sons, 1987.

[21] Brachman, R. J. and Schmolze, J. G., "An overview of the kl-one knowledge representation system," *Cognitive Science*, vol. 9, no. 2, pp. 171–216, 1985.

[22] BuildingSMART, *Industry Foundation Class (IFC) data model*, 2010. http://buildingsmart.com/standards/buildingsmart-standards/ifc (last accessed 10/23/2011).

[23] Caldas, C. and Soibelman, L., "Integration of construction documents in ifc project models," in *Proceedings of Construction Research Congress* (Molenaar, K. R. and Chinowsky, P. S., eds.), vol. 120, pp. 99–99, ASCE, 2003.

[24] Caldas, C., Soibelman, L., and others, "Methodology for the integration of project documents in model-based information systems," *Journal of computing in civil engineering*, vol. 19, p. 25, 2005.

[25] Cardelli, L. and Wegner, P., "On understanding types, data abstraction, and polymorphism," *ACM Comput. Surv.*, vol. 17, pp. 471–523, December 1985.

[26] Chandrasekaran, B., "Functional representation: A brief historical perspective," *Applied artificial intelligence*, vol. 8, no. 2, pp. 173–197, 1994.

[27] Chandrasekaran, B. and Milne, R., "Reasoning about structure, behavior and function," *SIGART Bull.*, pp. 4–55, July 1985.

[28] Charette, R. and Marshall, H., "Uniformat II Elemental Classifications for Building Specifications," *Cost Estimating, and Cost Analysis (NIST October 1999)*, 1999.

[29] Chidamber, S. and Kemerer, C., "A metrics suite for object oriented design," *Software Engineering, IEEE Transactions on*, vol. 20, pp. 476 –493, jun 1994.

[30] Chomsky, N., *Knowledge of language: Its nature, origin, and use*. Praeger Publishers, 1986.

[31] DBL, "Model view approach." 2011. (http://www.dbl.gatech.edu/ last accessed 20/23/2011).

[32] Eastman, C., Lee, J., Jeong, Y., and Lee, J., "Automatic rule-based checking of building designs," *Automation in Construction*, vol. 18, no. 8, pp. 1011–1033, 2009.

[33] Eastman, C., Panushev, I., Sacks, R., Venugopal, M., Aram, V., and See, R., "A guide for development and preparation of a national bim exchange standard," technical report, Georgia Tech and Technion, 2011.

[34] Eastman, C., Sacks, R., Panushev, I., Venugopal, M., and Aram, V., "Precast concrete bim standard documents:model view definitions for precast concrete," 2010.

[35] Eastman, C., Teicholz, P., Sacks, R., and Liston, K., *BIM Handbook: A guide to building information modeling for owners, managers, designers, engineers, and contractors*. John Wiley & Sons Inc, 2008.

[36] Eastman, C. M., Jeong, Y. S., Sacks, R., and Kaner, I., "Exchange model and exchange object concepts for implementation of national bim standards," *Journal of Computing in Civil Engineering*, vol. 24, no. 1, pp. 25–34, 2010.

[37] Eastman, C. M., Venugopal, M., and Aram, S., "Test implementation of interoperability standards for the precast/prestressed concrete industry (draft)," tech. rep., Digital Building Laboratory, Georgia Tech, 2012.

[38] Eastman, C., *Building product models: computer environments supporting design and construction*. CRC, 1999.

[39] Elmasri, R. and Navathe, S., *Fundamentals of database systems*, vol. 2. Pearson Education India, 2008.

[40] Farquhar, A., Fikes, R., and Rice, J., "The Ontolingua Server: a tool for collaborative ontology construction," *International Journal of Human-Computer Studies*, vol. 46, no. 6, pp. 707–727, 1997.

[41] Fensel, D., Horrocks, I., Van Harmelen, F., Decker, S., Erdmann, M., and Klein, M., "Oil in a nutshell," p. 1, Springer Verlag, 2000.

[42] Fischer, M. and Kunz, J., "The scope and role of information technology in construction," in *Proceedings - Japan Society of Civil Engineers*, pp. 1–32, Dotoku Gakkai, 2004.

[43] Gallagher, M., O'Connor, A., Dettbar, J., and Gilday, L., "Cost Analysis of Inadequate Interoperability in the US Capital Facilities Industry (NIST GCR 04-867)," 2004.

[44] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design patterns: elements of reusable object-oriented software*, vol. 206. Addison-wesley Reading, MA, 1995.

[45] Garrett, J., Fenves, S., and Stasiak, D., "A www-based regulation broker," *CIB REPORT*, pp. 219–230, 1996.

[46] Genesereth, M. and Fikes, R., "Knowledge interchange format, version 3.0 reference manual," 1992.

[47] Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubezy, M., Eriksson, H., Noy, N. F., and Tu, S. W., "The evolution of Protege: an environment for knowledge-based systems development," *International Journal of Human-Computer Studies*, vol. 58, no. 1, pp. 89–123, 2003.

[48] Goel, A., Rugaber, S., and Vattam, S., "Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 23, no. 01, pp. 23–35, 2009.

[49] Gómez-Pérez, A., Fernández-López, M., and Corcho, O., *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Verlag, 2004.

[50] Gruber, T. and others, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[51] Gruber, T. and others, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human Computer Studies*, vol. 43, no. 5, pp. 907–928, 1995.

[52] Gruber, T. and Olsen, G., "An ontology for engineering mathematics," in *Fourth International Conference on Principles of Knowledge Representation and Reasoning, Gustav Stresemann Institut, Bonn, Germany, Morgan Kaufmann*, pp. 241–245, 1994.

[53] Guarino, N., "Formal ontology, conceptual analysis and knowledge representation," *International Journal of Human Computer Studies*, vol. 43, no. 5-6, pp. 625–625, 1995.

[54] Guarino, N., "Formal ontology in information systems: Proceedings of the 1st international conference june 6-8, 1998, trento, italy," 1998.

[55] Guarino, N., Borgo, S., and Masolo, C., "Logical modelling of product knowledge: towards a well-founded semantics for step," in *Proceedingsof European Conference on Product Data Technology (PDT Days 97), Sophia Antipolis, France*, Citeseer, 1997.

[56] Guarino, N. and Welty, C., "A formal ontology of properties," *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, pp. 191–230, 2000.

[57] Gunter, C., *Semantics of programming languages: structures and techniques*. The MIT Press, 1992.

[58] Gunter, C. and Mitchell, J., *Theoretical aspects of object-oriented programming: types, semantics, and language design*. MIT Press, 1994.

[59] Haarslev, V. and Möller, R., "Racer: A core inference engine for the semantic web," in *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools*, vol. 87, Citeseer, 2003.

[60] Hietanen, J. and Final, S., "IFC model view definition format," *International Alliance for Interoperability*, 2006.

[61] Hjelseth, E. and Nisbet, N., "Exploring Semantic based Model Checking," in *CIB-W078 Conference in Cairo*, 2010.

[62] Hjelseth, E. and Nisbet, N., "Overview of concepts for model checking," in *CIB-W078 Conference in Cairo*, 2010.

[63] Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., and Wang, H., "The manchester owl syntax," *OWL: Experiences and Directions*, pp. 10–11, 2006.

[64] Horrocks, I., "Ontologies and the semantic web," *Commun. ACM*, vol. 51, no. 12, pp. 58–67, 2008.

[65] IABI, "Gtds- global testing documentation server." Institut for applied Building Informatics At University of Applied Sciences Munich, Faculty 02, 2011. gtds.buildingsmart.com/ (last accessed 10/23/2011).

[66] IAI, "Implementation guidelines for ifc 2x2 concrete domain," Tech. Rep. v1.0, IAI ST-3 PCC-IFC, 2004.

[67] IAI-Tech, "Industry foundation class (ifc) data model," tech. rep., BuildingSMART, 2003. http://buildingsmart-tech.org/specifications/ifc-releases/summary (last accessed 10/23/2011).

[68] ISO, "Standard data access interface," tech. rep., International Organization for Standardization, 1999. http://www.iso.org/iso/iso_catalogue/catalogue_tc/ (last accessed 10/23/2011).

[69] ISO, "ISO/PAS 16739: 2005," *Industry Foundation Classes*, 2005. http://www.iso.org/iso/ (last accessed 10/23/2011).

[70] Jeong, Y. S., Eastman, C. M., Sacks, R., and Kaner, I., "Benchmark tests for bim data exchanges of precast concrete," *Automation in Construction*, vol. 18, no. 4, pp. 469–484, 2009.

[71] Katranuschkov, P., Weise, M., Windisch, R., Fuchs, S., and Scherer, R., "BIM-based generation of Multi-Model Views," in *CIB-W078 Conference in Cairo*, 2010.

[72] Kifer, M. and Lausen, G., "F-logic: a higher-order language for reasoning about objects, inheritance, and scheme," *SIGMOD Rec.*, vol. 18, no. 2, pp. 134–146, 1989.

[73] Kiviniemi, A., *Requirements management interface to building product models*. Stanford University Stanford, CA, USA, 2005.

[74] Kiviniemi, A., Fischer, M., and Bazjanac, V., "Integration of multiple product models: Ifc model servers as a potential solution," in *Proc. of the 22nd CIB-W78 Conference on Information Technology in Construction*, 2005.

[75] Kiviniemi, A., "Ten Years of IFC Development- Why are we not yet there?," in *CIB-W78 Keynote*, (Montreal), 2007.

[76] Lassila, O. and McGuinness, D., "The role of frame-based representation on the semantic web," tech. rep., 2001.

[77] Lee, J., *Building Environment Rule and Analysis (BERA) Language*. PhD dissertation, Georgia Institute of Technology, College of Architecture, Dec.-Jan. 2010.

[78] Lee, J., Eastman, C., Lee, J., Kannala, M., and Jeong, Y., "Computing walking distances within buildings using the universal circulation network," *Environment and Planning B: Planning and Design*, vol. 37, no. 4, pp. 628–645, 2010.

[79] Liang, V. and Garrett, J., "Java-based regulation broker," *Journal of computing in civil engineering*, vol. 14, p. 100, 2000.

[80] Liebich, T., Adachi, Y., Forester, J., Hyvarinen, J., Karstila, K., and Wix, J., "Industry foundation classes ifc2x3," tech. rep., IAI, 2006.

[81] Liebich, T. and others, "Ifc 2x edition 2 model implementation guide," *International Alliance for Interoperability*, 2004.

[82] Liebich, T. and Wix, J., "Highlights of the development process of industry foundation classes," in *Proceedings of the second EC-PPM conference, Watford, UK*, 1998.

[83] Luke, S., Spector, L., Rager, D., and Hendler, J., "Ontology-based web agents," in *Proceedings of the first international conference on Autonomous agents*, pp. 59–66, ACM, 1997.

[84] Maedche, A., Motik, B., Stojanovic, L., Studer, R., and Volz, R., "Ontologies for enterprise knowledge management," *IEEE Intelligent Systems*, vol. 18, no. 2, pp. 26–33, 2003.

[85] Maile, T., Fischer, M., and Bazjanac, V., "Building energy performance simulation tools-a life-cycle and interoperable perspective," *Center for Integrated Facility Engineering (CIFE) Working Paper*, vol. 107, 2007.

[86] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A., "Wonderweb deliverable d18, ontology library (final)," tech. rep., 2009.

[87] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., and Schneider, L., "The WonderWeb library of foundational ontologies," *WonderWeb Deliverable D*, vol. 17, 2002.

[88] MasterFormat, C., "Construction Specifications Institute," *Alexandria, VA*, vol. 22314, 2010. http://www.masterformat.com/ (last accessed 10/23/2011).

[89] Milner, R., "A theory of type polymorphism in programming," *Journal of Computer and System Sciences*, vol. 17, no. 3, pp. 348 – 375, 1978.

[90] Musen, M. A., "Automated support for building and extending expert models," *Machine Learning*, vol. 4, no. 3, pp. 347–375, 1989.

[91] Navathe, S., "Evolution of data modeling for databases," *Communications of the ACM*, vol. 35, no. 9, pp. 112–123, 1992.

[92] NBIMS, "National bim standard overview, part -1," 2007. http://www.buildingsmartalliance.org/client/assets/files/bsa/ (last accessed 10/23/2011).

[93] NBIMS, "Ifc certification 2.0: Specification of certification process," 2010. http://buildingsmart-tech.org/certification (last accessed 05/21/2011).

[94] Nisbet, N. and Richter, S., "Repeated instances and placement sets," tech. rep., IAI, 2007.

[95] NIST, "Express engine project," 2009. http://exp-engine.sourceforge.net/ (last accessed 05/21/2011).

[96] Nour, M., "Performance of different (bim/ifc) exchange formats within private collaborative workspace for collaborative work," *ITcon*, vol. 14, no. Special Issue Building Information Modeling Applications, Challenges and Future Directions, pp. 736–752, 2009.

[97] Olofsson, T., Lee, G., and Eastman, C., "Editorial - case studies of bim in use," *ITcon*, vol. 13, no. Special Issue Case Studies of BIM in use, pp. 244–245, 2008.

[98] Owen, R., Amor, R., Palmer, M., Dickinson, J., Tatum, C., Kazi, A., Prins, M., Kiviniemi, A., and East, B., "Challenges for Integrated Design and Delivery Solutions," *Architectural Engineering and Design Management*, vol. 6, no. 4, pp. 232–240, 2010.

[99] Pahl, G., *Engineering design: a systematic approach*. Springer Verlag, 2007.

[100] Pazlar, T., Klinc, R., and Turk, Ž., "Mapping between architectural and structural aspects in the IFC based building information models," *EWork and EBusiness in Architecture, Engineering and Construction: ECPPM 2008*, p. 151, 2008.

[101] Pazlar, T. and Turk, Z., "Interoperability in practice: Geometric data exchange using the ifc standard," *ITcon*, vol. 13, p. 362, 2008.

[102] Peña-Mora, F., Anumba, C., Solari, J., and Duke, A., "An integrated telepresence environment for collaboration in construction," *Engineering with Computers*, vol. 16, pp. 287–305, 2000. 10.1007/PL00013717.

[103] Piegl, L., *Fundamental developments of computer-aided geometric modeling*. Academic Press, 1993.

[104] Rekola, M., Kojima, J., and Makelainen, T., "Towards integrated design and delivery solutions: Pinpointed challenges of process change," *Architectural Engineering and Design Management*, vol. 6, no. 4, pp. 264–278, 2010.

[105] Sacks, R., Eastman, C., Panushev, I., Venugopal, M., and Aram, V., "Precast Concrete BIM Standard Documents: IFC Extensions for Precast Concrete," *PCI-Charles Pankow Foundation. http://dcom. arch. gatech. edu/pcibim/documents/Precast (last accessed on 6/20/2010)*, 2010.

[106] Sacks, R., Kaner, I., Eastman, C. M., and Jeong, Y.-S., "The rosewood experiment – building information modeling and interoperability for architectural precast facades," *Automation in Construction*, vol. 19, no. 4, pp. 419–432, 2010.

[107] Schapke, S., Katranuschkov, P., and Scherer, R., "Towards Ontology-based Management of Distributed Multi-Modal Project Spaces," in *CIB-W078 Conference in Cairo*, 2010.

[108] Sembugamoorthy, V. and Chandrasekaran, B., "Functional representation of devices and compilation of diagnostic problem-solving systems," *Experience, memory and Reasoning*, pp. 47–73, 1986.

[109] Sheth, A. and Larson, J., "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Computing Surveys (CSUR)*, vol. 22, no. 3, pp. 183–236, 1990.

[110] Simons, P., *Parts: A study in ontology*. Oxford University Press, USA, 1987.

[111] Smith, I., *Intelligent computing in engineering and architecture*. Springer-Verlag GmbH., 2006.

[112] Stasiak, D., Garrett Jr, J., and Fenves, S., "A broker for tracking, delivering and using regulations over the world wide web," in *Electronics and the Environment, 1996. ISEE-1996., Proceedings of the 1996 IEEE International Symposium on*, pp. 293–297, IEEE, 1996.

[113] Strachey, C., *Towards a formal semantics*. North-Holland, 1966.

[114] Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D., "OntoEdit: Collaborative Ontology Development for the Semantic Web," *The Semantic Web-ISWC 2002, Springer*, pp. 221–235, 2002.

[115] Umeda, Y., Takeda, H., Tomiyama, T., and Yoshikawa, H., "Function, behaviour, and structure," *Applications of Artificial Intelligence in Engineering V*, vol. 1, pp. 177–193, 1990.

[116] Van Heijst, G., Schreiber, A., and Wielinga, B., "Using explicit ontologies in kbs development," *International Journal of Human Computer Studies*, vol. 46, pp. 183–292, 1997.

[117] Venugopal, M., Eastman, C., Sacks, R., Panushev, I., and Aram, V., "Engineering semantics of ifc product model views," in *Proceedings of the CIB W78 2010: 27 th International Conference –Cairo, Egypt, 16-18 November*, 2010.

[118] Venugopal, M., Eastman, C., Sacks, R., and Teizer, J., "Improving the robustness of model exchanges using product modeling" concepts" for ifc schema," in *Computing in Civil Engineering (2011)*, pp. 611–618, ASCE, 2011.

[119] Venugopal, M., Eastman, C. M., Sacks, R., and Teizer, J., "Semantics of model views for information exchanges using the industry foundation class schema," *Advanced Engineering Informatics, (in review)*, 2011.

[120] Venugopal, M., Eastman, C. M., and Teizer, J., "An ontology based approach for building model exchanges," *Advanced Engineering Informatics, (in review)*, 2011.

[121] Weise, M., Katranuschkov, P., and Scherer, R., "Generalised model subset definition schema," *CIB REPORT*, vol. 284, p. 440, 2003.

[122] Wix, J., Nisbet, N., and Liebich, T., "Using constraints to validate and check building information models," *eWork and eBusiness in Architecture, Engineering and Construction*, pp. 467–476, 2009.

[123] Yang, Q. and Zhang, Y., "Semantic interoperability in building design: Methods and tools," *Computer-Aided Design*, vol. 38, no. 10, pp. 1099–1112, 2006.

[124] Young, N., Jones, S., Bernstein, H., and Gudgel, J., "The Business Value of BIM: Getting Building Information Modeling to the Bottom Line," tech. rep., The McGraw-Hill Companies, New York, 2009.

VITA

Manu Venugopal received his Ph.D. from the School of Civil and Environmental Engineering at the Georgia Institute of Technology. His research areas includes Building Information Modeling, Interoperability, Process and Product Modeling, Virtual Design and Construction, Laser Scanning and Simulation. Manu completed his Bachelor of Engineering degree in Civil and Environmental Engineering from the National Institute of Technology (NIT), Karnataka, India, in 2005. From 2005 to 2007, he worked as a Systems Engineer at TATA Consultancy Services, India. He received Masters degrees in Civil Engineering and in Computational Science and Engineering from Georgia Institute of Technology. His doctoral research titled *Formal Specification of Industry Foundation Class Concepts using Engineering Ontologies* aims at improving the model exchange semantics and is funded by a National Institute of Standards and Technology (NIST) grant. At Georgia Tech, Manu has been involved in a variety of engineering projects involving Building Information Modeling (BIM) at the *Digital Building Lab* headed by Professor Chuck Eastman and emerging technologies such as Laser Scanning, Ultra Wideband and Simulations for the construction industry at the *RAPIDS Laboratory* headed by Professor Jochen Teizer. Manu played a leading role in the development of the Precast National BIM Standard in association with the Precast-Prestressed Concrete Institute (PCI) and the Charles Pankow Foundation. This work was recognized by the buildingSMART organization as one of the largest NBIMS activity till date. Some of his work is published in a research paper on *Semantics of Model Views for Information Exchanges using the Industry Foundation Class schema.*